

REFINED GENETIC ALGORITHMS
FOR
POLYPEPTIDE STRUCTURE PREDICTION

by

John D. Roberts, Ph.D.

Department of Biochemistry

University of California, Los Angeles

Los Angeles, California 90024

Submitted to the Graduate Committee in Biochemistry

in Partial Fulfillment of the Requirements for the Degree

of Doctor of Philosophy

Major Subject: Biochemistry

Minor Subject: Computer Science

Committee Chair: Dr. Robert L. Baldwin

Committee Member: Dr. James E. Johnson

Committee Member: Dr. John C. Tropsha

Committee Member: Dr. Michael J. Szwarc

Committee Member: Dr. David M. J. Lilley

Committee Member: Dr. James A. Dickey

Committee Member: Dr. James E. Johnson

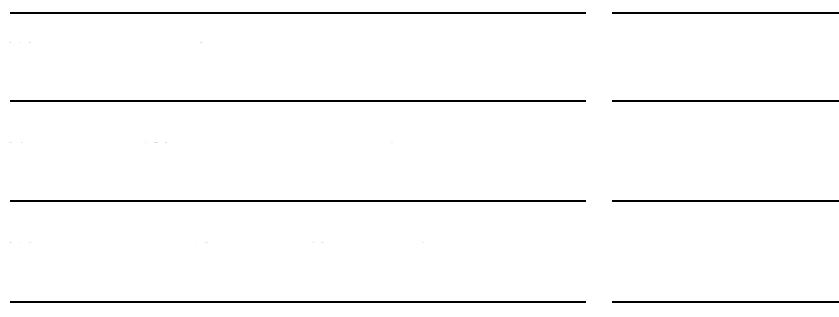
Committee Member: Dr. John C. Tropsha

Committee Member: Dr. Michael J. Szwarc

Committee Member: Dr. David M. J. Lilley

Committee Member: Dr. James A. Dickey

REFINED GENETIC ALGORITHMS
FOR
POLYPEPTIDE STRUCTURE PREDICTION



Acknowledgements

I would like to thank the editor and anonymous reviewers for their useful comments and suggestions which greatly improved this paper. I also thank Dr. Michael J. Lafferty for his support and encouragement. This research was partially funded by grants from the National Science Foundation (NSF) and the National Institute of Child Health and Human Development (NICHD). I am grateful to the NSF and NICHD for their support.

Correspondence concerning this article should be addressed to Dr. Ming Tang, Department of Sociology, University of California, Berkeley, CA 94720, USA. E-mail: tang@soc.berkeley.edu

Table of Contents

List of Figures

\times

P_s

List of Tables

ϕ, ψ

List of Symbols

| | |
|---------------|-----|
| α | ... |
| \mathcal{F} | ... |
| \mathcal{S} | ... |
| α | ... |
| γ | ... |
| i | ... |
| ϕ | ... |
| ψ | ... |
| ω | ... |
| χ_i | ... |
| i | ... |

List of Abbreviations

| | |
|----|-----------------|
| AB | alpha/beta |
| AC | alpha/carbon |
| AD | alpha/delta |
| AE | alpha/epsilon |
| AF | alpha/tau |
| AG | alpha/gamma |
| AK | alpha/kappa |
| AL | alpha/lambda |
| AN | alpha/nu |
| AP | alpha/phi |
| AS | alpha/sigma |
| AT | alpha/tau |
| AV | alpha/psi |
| AW | alpha/omega |
| BA | beta/alpha |
| BC | beta/carbon |
| BD | beta/delta |
| BE | beta/epsilon |
| BF | beta/tau |
| BG | beta/gamma |
| BK | beta/kappa |
| BL | beta/lambda |
| BN | beta/nu |
| BP | beta/phi |
| BS | beta/sigma |
| BT | beta/tau |
| BV | beta/psi |
| BW | beta/omega |
| CA | carbon/alpha |
| CD | carbon/delta |
| CE | carbon/epsilon |
| CF | carbon/tau |
| CG | carbon/gamma |
| CK | carbon/kappa |
| CL | carbon/lambda |
| CN | carbon/nu |
| CP | carbon/phi |
| CS | carbon/sigma |
| CT | carbon/tau |
| CV | carbon/psi |
| CW | carbon/omega |
| DA | delta/alpha |
| DC | delta/carbon |
| DD | delta/delta |
| DE | delta/epsilon |
| DF | delta/tau |
| DG | delta/gamma |
| DK | delta/kappa |
| DL | delta/lambda |
| DN | delta/nu |
| DP | delta/phi |
| DS | delta/sigma |
| DT | delta/tau |
| DV | delta/psi |
| DW | delta/omega |
| EA | epsilon/alpha |
| EC | epsilon/carbon |
| ED | epsilon/delta |
| EE | epsilon/epsilon |
| EF | epsilon/tau |
| EG | epsilon/gamma |
| EK | epsilon/kappa |
| EL | epsilon/lambda |
| EN | epsilon/nu |
| EP | epsilon/phi |
| ES | epsilon/sigma |
| ET | epsilon/tau |
| EV | epsilon/psi |
| EW | epsilon/omega |
| FA | tau/alpha |
| FC | tau/carbon |
| FD | tau/delta |
| FE | tau/epsilon |
| FF | tau/tau |
| FG | tau/gamma |
| FK | tau/kappa |
| FL | tau/lambda |
| FN | tau/nu |
| FP | tau/phi |
| FS | tau/sigma |
| FT | tau/tau |
| FV | tau/psi |
| FW | tau/omega |
| GA | gamma/alpha |
| GC | gamma/carbon |
| GD | gamma/delta |
| GE | gamma/epsilon |
| GF | gamma/tau |
| GG | gamma/gamma |
| GK | gamma/kappa |
| GL | gamma/lambda |
| GN | gamma/nu |
| GP | gamma/phi |
| GS | gamma/sigma |
| GT | gamma/tau |
| GV | gamma/psi |
| GW | gamma/omega |
| KA | kappa/alpha |
| KC | kappa/carbon |
| KD | kappa/delta |
| KE | kappa/epsilon |
| KF | kappa/tau |
| KG | kappa/gamma |
| KK | kappa/kappa |
| KL | kappa/lambda |
| KN | kappa/nu |
| KP | kappa/phi |
| KS | kappa/sigma |
| KT | kappa/tau |
| KV | kappa/psi |
| KW | kappa/omega |
| LA | lambda/alpha |
| LC | lambda/carbon |
| LD | lambda/delta |
| LE | lambda/epsilon |
| LF | lambda/tau |
| LG | lambda/gamma |
| LK | lambda/kappa |
| LL | lambda/lambda |
| LN | lambda/nu |
| LP | lambda/phi |
| LS | lambda/sigma |
| LT | lambda/tau |
| LV | lambda/psi |
| LW | lambda/omega |
| NA | nu/alpha |
| NC | nu/carbon |
| ND | nu/delta |
| NE | nu/epsilon |
| NF | nu/tau |
| NG | nu/gamma |
| NK | nu/kappa |
| NL | nu/lambda |
| NN | nu/nu |
| NP | nu/phi |
| NS | nu/sigma |
| NT | nu/tau |
| NV | nu/psi |
| NW | nu/omega |
| PA | phi/alpha |
| PC | phi/carbon |
| PD | phi/delta |
| PE | phi/epsilon |
| PF | phi/tau |
| PG | phi/gamma |
| PK | phi/kappa |
| PL | phi/lambda |
| PN | phi/nu |
| PP | phi/phi |
| PS | phi/sigma |
| PT | phi/tau |
| PV | phi/psi |
| PW | phi/omega |
| SA | sigma/alpha |
| SC | sigma/carbon |
| SD | sigma/delta |
| SE | sigma/epsilon |
| SF | sigma/tau |
| SG | sigma/gamma |
| SK | sigma/kappa |
| SL | sigma/lambda |
| SN | sigma/nu |
| SP | sigma/phi |
| SS | sigma/sigma |
| ST | sigma/tau |
| SV | sigma/psi |
| SW | sigma/omega |
| TA | tau/alpha |
| TC | tau/carbon |
| TD | tau/delta |
| TE | tau/epsilon |
| TF | tau/tau |
| TG | tau/gamma |
| TK | tau/kappa |
| TL | tau/lambda |
| TN | tau/nu |
| TP | tau/phi |
| TS | tau/sigma |
| TT | tau/tau |
| TV | tau/psi |
| TW | tau/omega |
| ZA | omega/alpha |
| ZC | omega/carbon |
| ZD | omega/delta |
| ZE | omega/epsilon |
| ZF | omega/tau |
| ZG | omega/gamma |
| ZK | omega/kappa |
| ZL | omega/lambda |
| ZN | omega/nu |
| ZP | omega/phi |
| ZS | omega/sigma |
| ZT | omega/tau |
| ZV | omega/psi |
| ZW | omega/omega |

Abstract

REFINED GENETIC ALGORITHMS
FOR
POLYPEPTIDE STRUCTURE PREDICTION

I. Introduction

The protein folding problem (PFP) is one of the most important problems in molecular biology. It consists of predicting the native conformation of a polypeptide chain from its primary sequence. The native conformation is the unique three-dimensional structure that a protein assumes in its natural environment. It is responsible for the protein's biological functions and is determined by the sequence of amino acids and the interactions between them. The PFP is a complex problem because it involves the study of the physical and chemical properties of proteins, their interactions with other molecules, and the effects of environmental factors such as temperature, pH, and solvent.

There are several approaches to solving the PFP, including experimental methods and computational methods. Experimental methods involve physically unfolding a protein and then refolding it under different conditions to determine its native conformation. Computational methods, on the other hand, involve using mathematical models and algorithms to predict the native conformation based on the protein's sequence. One such computational method is the refined genetic algorithm (RGA), which has been shown to be effective in solving the PFP.

The RGA is a type of optimization algorithm that uses the principles of natural selection and genetics to search for the best solution. It starts with a population of initial solutions, called individuals, which are represented as binary strings. These individuals are evaluated based on a fitness function that measures how well they satisfy the constraints of the PFP. The individuals with the highest fitness are selected to produce the next generation through crossover and mutation operations. This process is repeated until a satisfactory solution is found or a maximum number of generations is reached.

The RGA has been used to solve various instances of the PFP, including the prediction of the native conformations of small proteins and peptides. It has been shown to be able to find solutions that are comparable in quality to those obtained by other methods, while being more efficient in terms of computation time and memory usage. In addition, the RGA can handle large-scale problems involving thousands of amino acids, which is a significant advantage over other methods.

In this paper, we present a refined version of the RGA for polypeptide structure prediction. The refinement involves several key improvements, such as a more accurate fitness function, better crossover and mutation operators, and a more efficient search strategy. We also compare the performance of the refined RGA with other state-of-the-art methods on a set of benchmark problems. The results show that the refined RGA is able to find solutions that are competitive with the best methods available, while being more robust and easier to implement.

1.1 Protein Folding Problem / Polypeptide Structure Prediction

The protein folding problem (PFP) is one of the most important problems in molecular biology. It consists of predicting the native conformation of a polypeptide chain from its primary sequence. The native conformation is the unique three-dimensional structure that a protein assumes in its natural environment. It is responsible for the protein's biological functions and is determined by the sequence of amino acids and the interactions between them. The PFP is a complex problem because it involves the study of the physical and chemical properties of proteins, their interactions with other molecules, and the effects of environmental factors such as temperature, pH, and solvent.

There are several approaches to solving the PFP, including experimental methods and computational methods. Experimental methods involve physically unfolding a protein and then refolding it under different conditions to determine its native conformation. Computational methods, on the other hand, involve using mathematical models and algorithms to predict the native conformation based on the protein's sequence. One such computational method is the refined genetic algorithm (RGA), which has been shown to be effective in solving the PFP.

The RGA is a type of optimization algorithm that uses the principles of natural selection and genetics to search for the best solution. It starts with a population of initial solutions, called individuals, which are represented as binary strings. These individuals are evaluated based on a fitness function that measures how well they satisfy the constraints of the PFP. The individuals with the highest fitness are selected to produce the next generation through crossover and mutation operations. This process is repeated until a satisfactory solution is found or a maximum number of generations is reached.

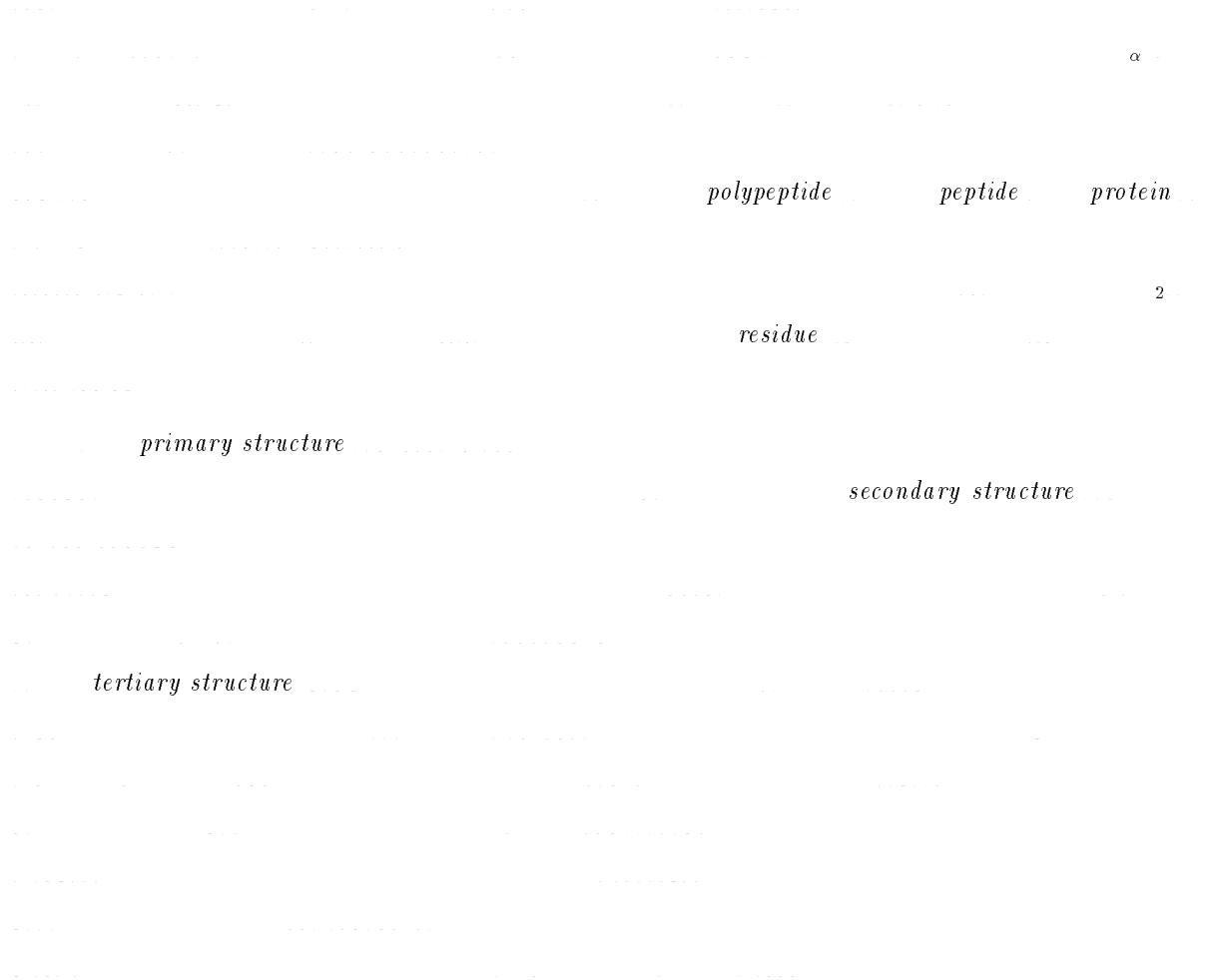
The RGA has been used to solve various instances of the PFP, including the prediction of the native conformations of small proteins and peptides. It has been shown to be able to find solutions that are comparable in quality to those obtained by other methods, while being more efficient in terms of computation time and memory usage. In addition, the RGA can handle large-scale problems involving thousands of amino acids, which is a significant advantage over other methods.

In this paper, we present a refined version of the RGA for polypeptide structure prediction. The refinement involves several key improvements, such as a more accurate fitness function, better crossover and mutation operators, and a more efficient search strategy. We also compare the performance of the refined RGA with other state-of-the-art methods on a set of benchmark problems. The results show that the refined RGA is able to find solutions that are competitive with the best methods available, while being more robust and easier to implement.

¹The *native conformation* determines the protein's biological functions.

$$PSP \subseteq PFP$$

1.1.1 Background.



1.1.2 Importance.



1.1.3 Methods for Polypeptide Structures Prediction.

energy minimization molecular dynamics

ab initio

semi-empirical

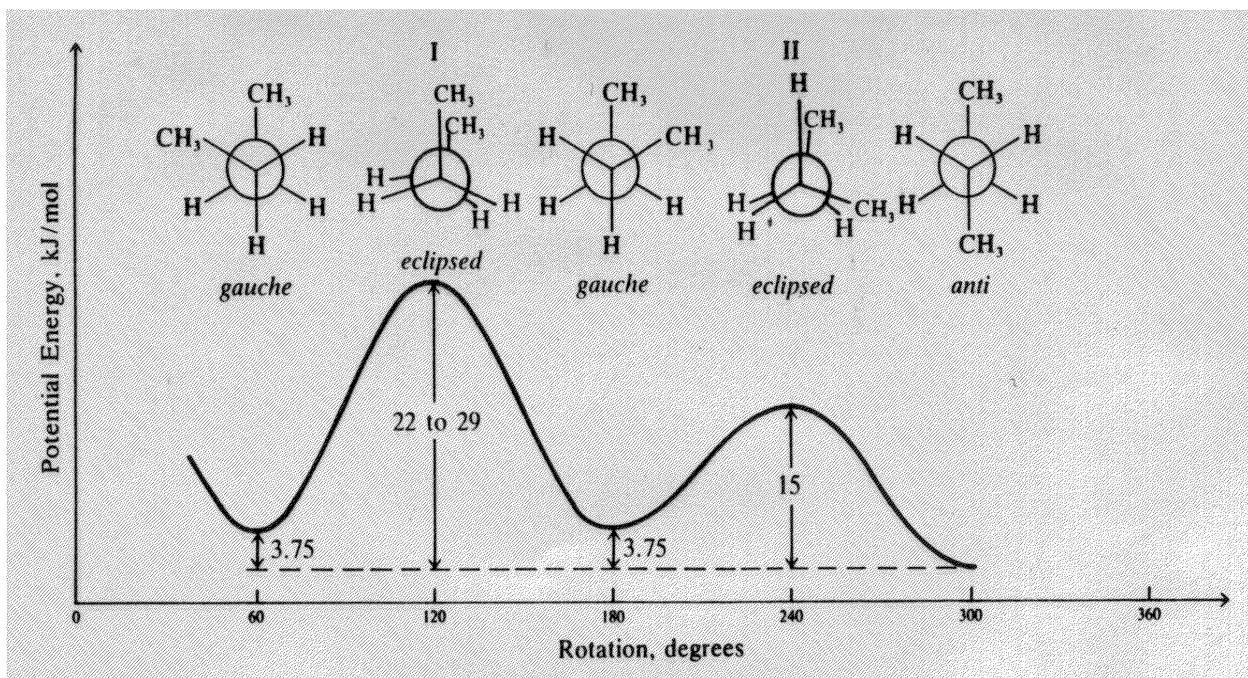
force-field

$\mathcal{O} n^5$

$\mathcal{O} n^4$

$\mathcal{O} n^2$

Time (ns) n (number of atoms)



n

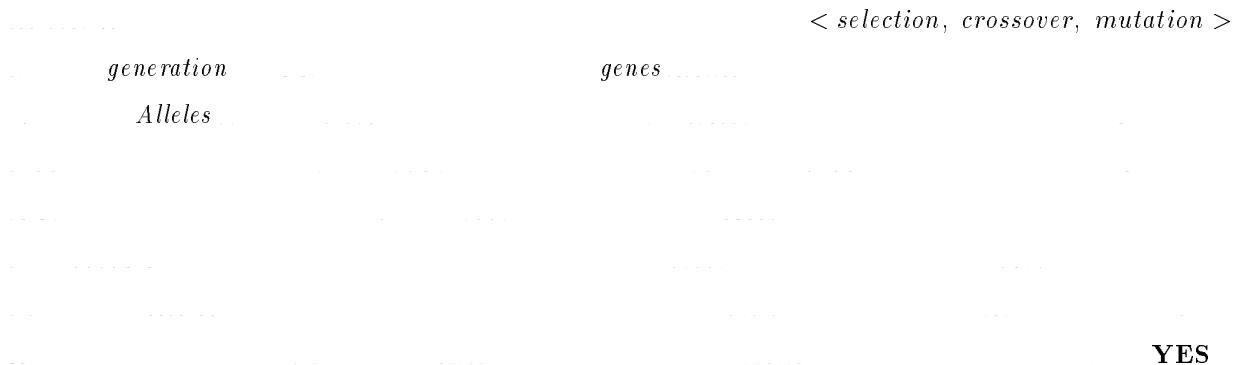
1.1.4 Growth of Complexity.

1.2 Genetic Algorithms

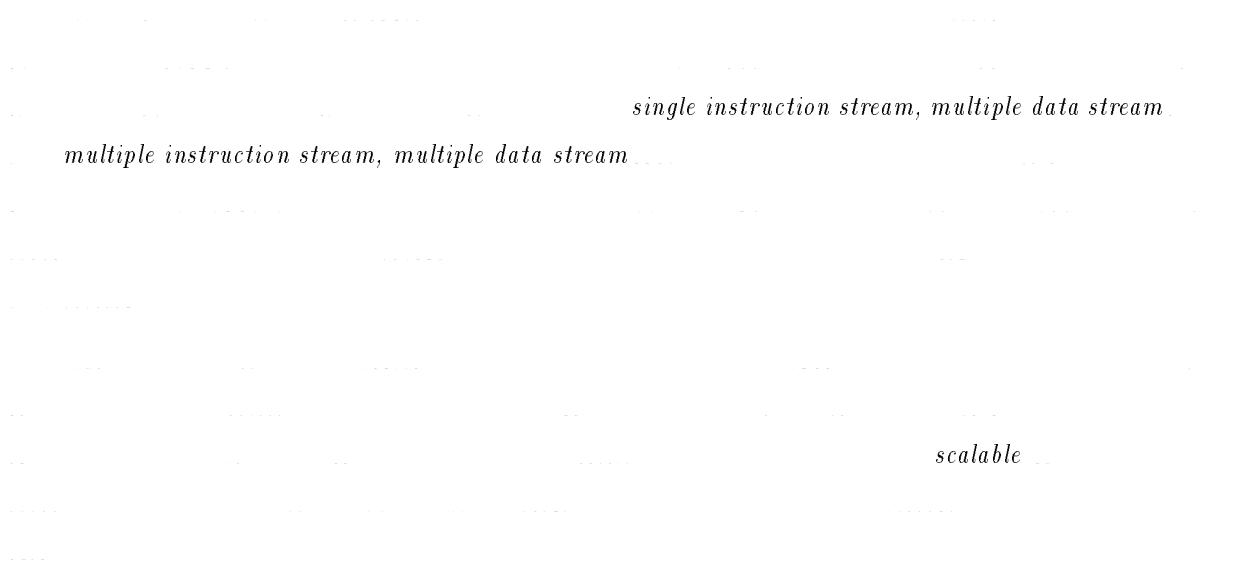
² Regardless of whether a *Cartesian* or *Internal* coordinate system is used. However, the internal coordinate system has fewer independent variables.

³See Appendix B.3.3

⁴Because GAs are loosely based on natural evolution, many of the terms associated with natural evolution are used interchangeably with the terms created specifically for genetic algorithms (67).



1.3 Parallel and Distributed Computing

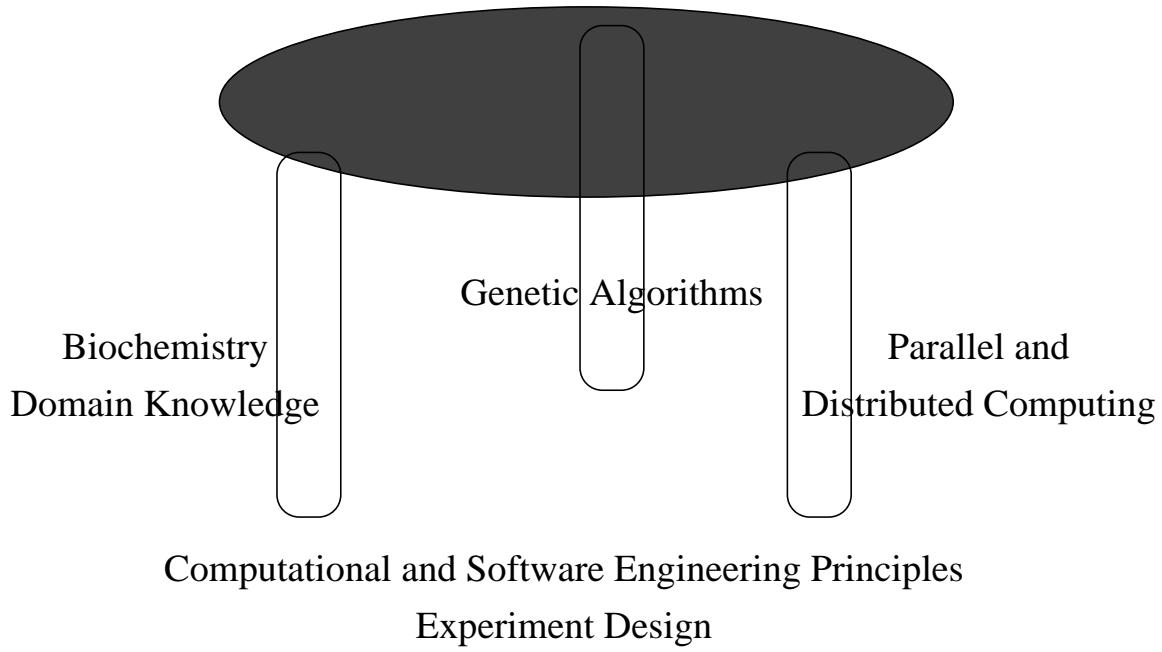


1.4 Research Objectives



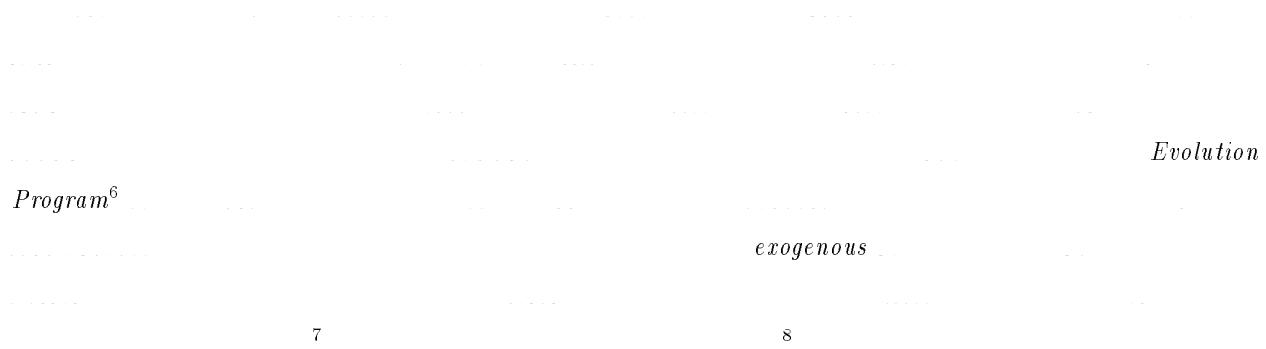
⁵Crossover is sometimes called recombination.

Effective and Efficient Polypeptide Structure Prediction



- Improve Performance of Hybrid GAs for PSP
- Real Valued Genetic Algorithm Implementation for the PSP
- Exploit Domain Knowledge to Limit Search Space
 - domain knowledge

1.5 Methodology



1.6 Assumptions



1.7 Summary



⁶Not to be confused with *Evolutionary Programming*, see Section 2.4.

⁷The probability that an improvement at a specific node is migrated to other nodes.

⁸The probability, given a migration, that it is migrated to all other nodes.

⁹Or molecular conformation

II. Current Issues

2.1 Introduction

2.2 Previous Research

2.3 Polypeptides Structure Prediction (PSP)



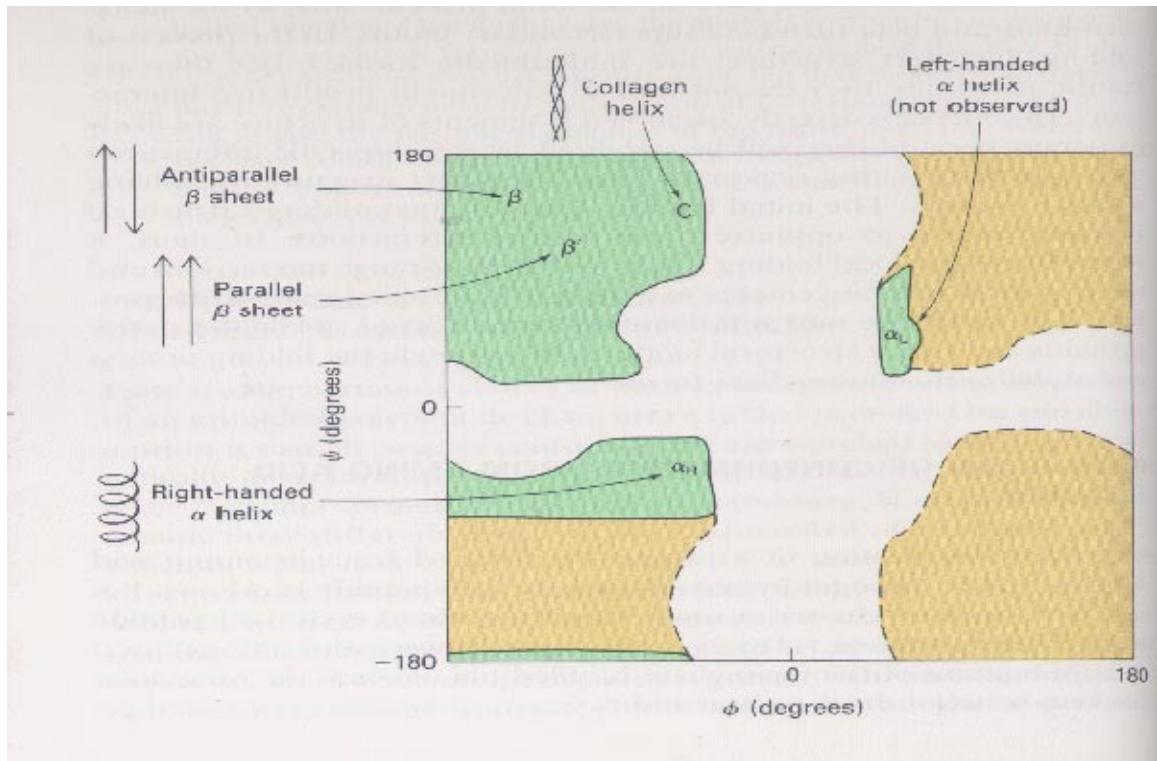
¹⁰ See also the discussion of the relationship between *trans*- and *intra*-cultural communication in the section on "Cultural Transmissions" below.

ϕ, ψ, ω are the three parameters of the model.

Ramachandran Plot

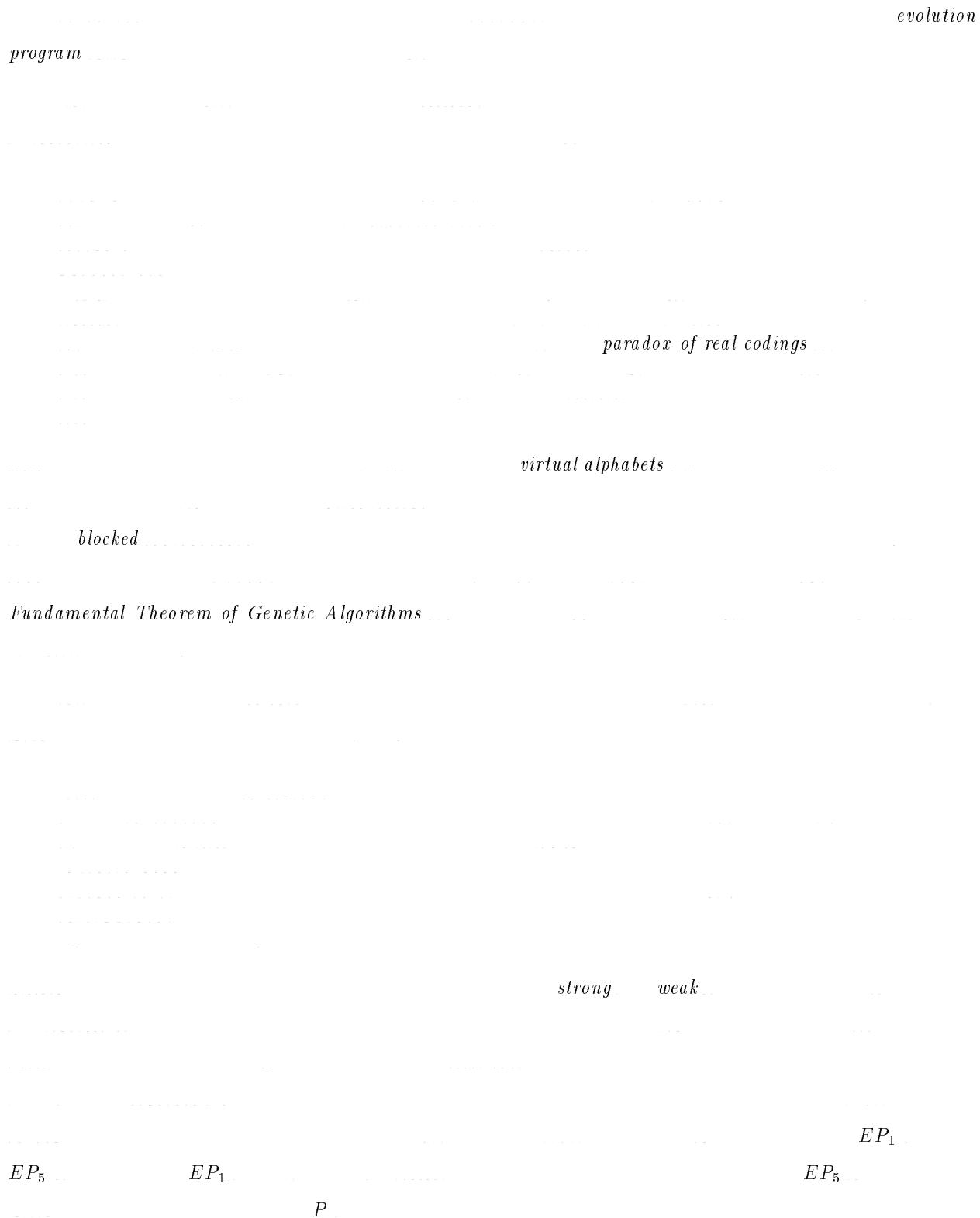
ψ

ϕ



Biochemistry

2.4 Genetic Algorithms



2.5 Parallel Genetic Algorithms

Parallel Genetic Algorithms (PGAs) have been developed to take advantage of parallel processing power. In a PGA, the population is divided into smaller subpopulations, each of which is evolved independently. This allows for a more efficient search of the solution space, as multiple subpopulations can explore different regions simultaneously. However, it also requires careful coordination between the subpopulations to ensure that they do not get stuck in local optima or compete for resources. One common approach is to use a "migration" mechanism, where individuals from one subpopulation are periodically moved to another to promote genetic diversity and prevent premature convergence. Another approach is to use a "cooperative" mechanism, where subpopulations share information and coordinate their evolution to reach a global optimum. Overall, PGAs offer a promising way to solve complex optimization problems by leveraging the power of parallel computing.



Island Model:

migration

$$\mathcal{O}\left(\frac{nl}{p}\right)$$

$p \ll n$

$$p = \frac{n}{l}$$

$$n$$

Neighborhood Model:

$$n = p$$

$$\mathcal{O}(s - l) = \mathcal{O}(nl)$$

$$s$$

Farming Model:

farming

farm out

foreman

workers

2.6 Summary

III. Algorithm Analysis, Design, and Implementation

Algorithm analysis, design, and implementation are critical components of scientific computing. In this section, we will discuss the analysis of local minimization using conjugate gradient, the design of Lamarckian and Baldwinian optimization algorithms, and the implementation of generational checkpointing and asynchronous farming.

3.1 Analysis

3.1.1 Cost Analysis of Local Minimization using Conjugate Gradient.

The cost of local minimization using conjugate gradient is proportional to the number of iterations required to reach a minimum. This cost can be analyzed by examining the convergence behavior of the algorithm.

Numerical Recipes in C

The cost of local minimization using conjugate gradient is proportional to the number of iterations required to reach a minimum. This cost can be analyzed by examining the convergence behavior of the algorithm.

The cost of local minimization using conjugate gradient is proportional to the number of iterations required to reach a minimum. This cost can be analyzed by examining the convergence behavior of the algorithm.

$\mathcal{O}(n^2)$ **ITMAX = 200**

The cost of local minimization using conjugate gradient is proportional to the number of iterations required to reach a minimum. This cost can be analyzed by examining the convergence behavior of the algorithm.

Lamarckian Baldwinian

The cost of local minimization using conjugate gradient is proportional to the number of iterations required to reach a minimum. This cost can be analyzed by examining the convergence behavior of the algorithm.

```
eval_func()  
ITMAX
```

The cost of local minimization using conjugate gradient is proportional to the number of iterations required to reach a minimum. This cost can be analyzed by examining the convergence behavior of the algorithm.

The cost of local minimization using conjugate gradient is proportional to the number of iterations required to reach a minimum. This cost can be analyzed by examining the convergence behavior of the algorithm.

The cost of local minimization using conjugate gradient is proportional to the number of iterations required to reach a minimum. This cost can be analyzed by examining the convergence behavior of the algorithm.

The cost of local minimization using conjugate gradient is proportional to the number of iterations required to reach a minimum. This cost can be analyzed by examining the convergence behavior of the algorithm.

The cost of local minimization using conjugate gradient is proportional to the number of iterations required to reach a minimum. This cost can be analyzed by examining the convergence behavior of the algorithm.

The cost of local minimization using conjugate gradient is proportional to the number of iterations required to reach a minimum. This cost can be analyzed by examining the convergence behavior of the algorithm.

The cost of local minimization using conjugate gradient is proportional to the number of iterations required to reach a minimum. This cost can be analyzed by examining the convergence behavior of the algorithm.

The cost of local minimization using conjugate gradient is proportional to the number of iterations required to reach a minimum. This cost can be analyzed by examining the convergence behavior of the algorithm.

The cost of local minimization using conjugate gradient is proportional to the number of iterations required to reach a minimum. This cost can be analyzed by examining the convergence behavior of the algorithm.

The cost of local minimization using conjugate gradient is proportional to the number of iterations required to reach a minimum. This cost can be analyzed by examining the convergence behavior of the algorithm.

The cost of local minimization using conjugate gradient is proportional to the number of iterations required to reach a minimum. This cost can be analyzed by examining the convergence behavior of the algorithm.

client ...
... *server* ...

... *client* ...
... *server* ...

| | | | | | |
|--|--|--|--|--|--|
| | | | | | |
| | | | | | |
| | | | | | |
| | | | | | |

charm_eval()

... *client* ...
... *server* ...

charm_eval()

... *client* ...
... *server* ...

charm_eval()

... *client* ...
... *server* ...

the backbone atoms. The backbone atoms are the carbonyl carbon, the alpha-carbon, and the carbons of the amide group.

3.1.2 Constraint Set Development.

The constraint set development process involves defining the range of possible conformations for each bond angle in the molecule. This is done by specifying the minimum and maximum allowed values for each bond angle, and then determining the range of conformations that satisfy these constraints.

3.1.2.1 Conventions Adopted.

The conventions adopted for bond angles are as follows:

- θ_{min} and θ_{max} represent the minimum and maximum allowed values for a bond angle, respectively.
- $\theta_{min_{adj}}$ and $\theta_{max_{adj}}$ represent the minimum and maximum allowed values for an adjacent bond angle, respectively.
- $\theta_{min} < \theta_{max} \rightarrow \theta_{min_{adj}} = \theta_{min}$
- $\theta_{min} > \theta_{max} \rightarrow \theta_{max_{adj}} = \theta_{max}$
- $\phi\psi$ and α represent the torsion angles between the C₁-C₂-C₃-C₄ and C₂-C₃-C₄-C₅ planes, respectively.

3.1.2.2 [Met]-enkephalin.

The bond angles for the [Met]-enkephalin molecule are defined as follows:

| | |
|------------|----------|
| $\phi\psi$ | α |
| $\phi\psi$ | α |

3.1.2.3 Polyalanine.

The bond angles for the polyalanine molecule are defined as follows:

| | |
|------------|----------|
| $\phi\psi$ | α |
| $\phi\psi$ | α |

| | | | | θ_{min} | θ_{max} |
|--------------------|---|--|---|----------------|----------------|
| <i>Non-glycine</i> | — | | | — | — |
| <i>Glycine</i> | — | | | — | — |
| | | | | — | — |
| | — | | | — | — |
| χ_1 | — | | — | — — | — — |

| | | | | θ_{min} | θ_{max} |
|--------------------|---|---|---------------------|---------------------|----------------|
| <i>Non-glycine</i> | — | | | — | |
| <i>Glycine</i> | | | | — | — |
| | | | | — | — |
| | | . | . | — | — . |
| χ_1 | — | . | — — | — . . . — . | |

$$\alpha$$

| | | | | θ_{min} | θ_{max} |
|----------|-----|---|---|----------------|----------------|
| | — . | . | | — | — |
| | — | | | — | — |
| | | | | — | — |
| χ_1 | — | | — | — — | — — |

3.1.3 Real-valued GAs.

| | | | | θ_{min} | θ_{max} |
|----------|---|--|--|----------------|----------------|
| | — | | | — | — |
| | — | | | — | — |
| | | | | — | — |
| χ_1 | — | | | — | — |

1

Figure 3.1: A schematic diagram of the parameter space search space for the first parameter χ_1 . The search space is divided into three regions by two vertical lines. The first region is bounded by the left boundary and the first vertical line, the second region is bounded by the first and second vertical lines, and the third region is bounded by the second and right boundaries. The horizontal axis represents the value of χ_1 , and the vertical axis represents the probability density function.

The search space is divided into three regions by two vertical lines. The first region is bounded by the left boundary and the first vertical line, the second region is bounded by the first and second vertical lines, and the third region is bounded by the second and right boundaries.

3.2 Algorithm Design and Implementation

The algorithm design and implementation process involves several steps. First, the problem is defined and the search space is determined. Then, the search space is divided into regions based on the parameter values. Next, the search space is explored using a parallel hybrid genetic algorithm. Finally, the results are analyzed and the model is validated.

3.2.1 Parallel Hybrid GA.

The parallel hybrid genetic algorithm (GA) is a combination of a traditional GA and a parallel search algorithm. It uses a population of individuals to search for the optimal solution. The search space is divided into regions, and each region is assigned to a different processor. The processors work in parallel to explore the search space. The results are then combined to find the global optimum.

3.2.1.1 Algorithm Design.

The algorithm design process involves several steps. First, the problem is defined and the search space is determined. Then, the search space is divided into regions based on the parameter values. Next, the search space is explored using a parallel hybrid genetic algorithm. Finally, the results are analyzed and the model is validated.

The search space is divided into regions by two vertical lines. The first region is bounded by the left boundary and the first vertical line, the second region is bounded by the first and second vertical lines, and the third region is bounded by the second and right boundaries.

The search space is divided into regions by two vertical lines. The first region is bounded by the left boundary and the first vertical line, the second region is bounded by the first and second vertical lines, and the third region is bounded by the second and right boundaries.

The search space is divided into regions by two vertical lines. The first region is bounded by the left boundary and the first vertical line, the second region is bounded by the first and second vertical lines, and the third region is bounded by the second and right boundaries.

The search space is divided into regions by two vertical lines. The first region is bounded by the left boundary and the first vertical line, the second region is bounded by the first and second vertical lines, and the third region is bounded by the second and right boundaries.

The search space is divided into regions by two vertical lines. The first region is bounded by the left boundary and the first vertical line, the second region is bounded by the first and second vertical lines, and the third region is bounded by the second and right boundaries.

The search space is divided into regions by two vertical lines. The first region is bounded by the left boundary and the first vertical line, the second region is bounded by the first and second vertical lines, and the third region is bounded by the second and right boundaries.

¹The developers of this algorithm have been inconsistent with its naming in the literature. Genocop, Genocop-III.1.0, GENOCOP-III, etc., have been used synonymously. In this document I have adopted the standardization of GENOCOP-III.

Q
 $EvalCnt \leftarrow$ population size
Loop

Until $EvalCnt$
Loop

$EvalCnt \leftarrow$
Loop

Until $EvalCnt$
Until

3.2.1.2 Scheduling.

round robin

network of workstations

3.2.2 Real-valued GA, Limited by constraints (REGAL)

evolution program

Loop

Until

REPLACE

| | |
|---|----------------------------------|
| $server_i \quad Q$ $EvalsPerformed$ $EvalsPerformed < EvalCnt \wedge Q / \emptyset$ | $server_j \quad Q$ $server_j$ |
|---|----------------------------------|

3.2.2.1 Incorporation of Domain Knowledge.

Evolution Program

probable improbable \mathcal{S}_{prob}

\mathcal{S}_{improb}

$\mathcal{S}_{prob} \quad \mathcal{S}_{improb} \quad \mathcal{S}$

$\mathcal{S}_{prob} \quad \mathcal{S}_{improb} \quad \emptyset$

$$x_i \in \vec{x} \in R$$

$$\theta_{i_{min}} \quad \theta_{i_{max}} \quad x_i \in \vec{x} \in R$$

$$\theta_{i_{min}} \quad \theta_{i_{max}} \quad -\pi \quad \pi$$

$$\leq \theta - \frac{\theta_{min} - \theta_{max}}{\theta_{max} - \theta_{min}}$$

$$\{\phi, \psi, \omega\}$$

$$\leq \theta - \frac{\theta_{min} - \theta_{max}}{\theta_{max} - \theta_{min}}$$

$$\{\chi_1\}$$

3.2.3 Parallel REGAL (Para-REGAL).

modified island

island

2

probabilistic migration

Probability of Migration

P_m *Probability of Complete Migration* P_{cm}

$P_m \cdot P_{cm}$

i j

$P_m \cdot P_{cm}$

$P_m \cdot P_{cm} >$

*archipelago*³

- Average Genotypic Distance

- Least Genotypic Distance

²Except for the seed of the random number generator.

³A collection of islands; thus the islands making of the Para-REGAL execution

- *Greatest Genotypic Distance*
- *Average Best Fitness*
- *Local Delta*

3.3 Summary

The three metrics used to measure the performance of the genetic algorithm are the greatest genotypic distance, average best fitness, and local delta. The greatest genotypic distance measures the difference between the best and worst individuals in the population. The average best fitness measures the average fitness of the best individuals in the population. The local delta measures the change in the average best fitness over time. These metrics provide a way to evaluate the performance of the genetic algorithm and make adjustments to the parameters if necessary.

IV. Experiment Design

The experiment design is divided into two parts. The first part is to validate the model by comparing the simulation results with the experimental data. The second part is to study the effect of different parameters on the system performance. The validation part consists of three steps: (1) setting up the simulation environment; (2) running the simulation and collecting data; (3) comparing the simulation results with the experimental data. The validation part is completed successfully. The study part consists of four steps: (1) identifying the parameters to be studied; (2) setting up the simulation environment; (3) running the simulation and collecting data; (4) analyzing the data and drawing conclusions. The study part is also completed successfully.

4.1 Experiment Techniques

The experiment techniques used in this study include:
1. Simulation: A computer program is used to simulate the system behavior under different conditions. The simulation results are compared with the experimental data to validate the model.
2. Experimental: Actual experiments are conducted on the system to collect data. The experimental data is used to validate the simulation results and to study the effect of different parameters on the system performance.

4.1.1 Random Number Seeds.

The random number seeds used in the simulation are generated by a linear congruential generator. The generator uses the following formula to generate the next seed value:

nodal

```
Seed = (unsigned) ((Seed + My_node) / (My_node + 1));
```

```
Seed =(unsigned) ((Seed + My_node) % Max_Seed_Value);
```

Seed = floor(*rand number from Xcalculator)

Seed = floor(*random number from Xcalculator)
Seed = floor(*random number from Xcalculator)

| | | | |
|--|--|--|--|
| | | | |
| | | | |
| | | | |
| | | | |
| | | | |
| | | | |
| | | | |
| | | | |

4.1.2 Statistical Techniques.

4.1.2.1 Analysis of Variance (ANOVA).

$$\sigma^2$$

4.1.2.2 Kruskal-Wallis H Test.

The Kruskal-Wallis H test is a non-parametric statistical test used to determine if there are differences between two or more independent groups. It is similar to the one-way ANOVA, but it does not assume normality of the data. The null hypothesis for the Kruskal-Wallis H test is that all groups have the same median. If the null hypothesis is rejected, then the groups are considered to be significantly different. The H statistic is calculated as follows:

$$H = \frac{12}{N(N+1)} \sum_{i=1}^k \frac{R_i^2}{n_i} - 3(N+1)$$

where N is the total number of observations, k is the number of groups, n_i is the number of observations in group i , and R_i is the rank sum for group i . The p-value is then determined based on the H statistic and the degrees of freedom.

4.2 Experiment I: Evaluation of the Efficiency of a Parallel & Distributed Hybrid GA

4.2.1 Motivation and Objective.

The motivation for this experiment is to evaluate the efficiency of a parallel and distributed hybrid Genetic Algorithm (GA). The objective is to compare the performance of the proposed hybrid GA with other existing GAs. The hybrid GA combines the strengths of both parallel and distributed GAs. It uses a population-based search space and a distributed search space. The parallel component is responsible for performing local search operations on the population. The distributed component is responsible for performing global search operations on the distributed search space. The hybrid GA is expected to be more efficient than the traditional GAs because it can explore a larger search space and find better solutions faster. The performance of the hybrid GA will be evaluated based on various metrics such as convergence rate, solution quality, and computational cost.

4.2.2 Methodology.

The methodology for this experiment involves the following steps:

1. Problem Definition: The problem to be solved is a complex optimization problem with multiple local optima. The problem is divided into several subproblems, which are assigned to different nodes in the distributed search space.
2. Algorithm Design: The hybrid GA is designed to handle the distributed search space. It uses a population-based search space and a distributed search space. The parallel component performs local search operations on the population. The distributed component performs global search operations on the distributed search space.
3. Implementation: The hybrid GA is implemented using a distributed computing framework. The distributed search space is implemented using a peer-to-peer network. The parallel component is implemented using a shared memory model. The distributed component is implemented using a message-passing model.
4. Performance Evaluation: The performance of the hybrid GA is evaluated based on various metrics such as convergence rate, solution quality, and computational cost. The results are compared with other existing GAs.

987654321

4.3 Experiment II: Evaluation of the Use of Constraints in the PSP

4.3.1 Motivation and Objective.

4.3.2 Methodology.

none *loose* *tight*

[Met]-enkephalin tight Polyalanine tight

4.3.3 Parameter Selection.

reference population

F S

1

loose *tight*

4.4 Experiment III: Evaluation of Exogenous Parameters in the REGAL System

4.4.1 Motivation and Objective.

¹ Based on the ratio of $\frac{F}{S}$, it would require 10^{67} tries to randomly generate just one fully feasible chromosome when using the tight constraint set for Polyalanine.

4.4.2 Methodology.

steady state \rightarrow $\text{[A]}/\text{[A}_0 = \text{[B]}/\text{[B}_0 = \text{[C]}/\text{[C}_0 = \dots = k_2/k_1$

adaptive

4.4.3 Exogenous Parameter Evaluation Experiments.

Offsprings *Reference Population Size*

| | | | |
|-----|--|--|--|
| 1 | | | |
| 2 | | | |
| 3 | | | |
| 4 | | | |
| 5 | | | |
| 6 | | | |
| 7 | | | |
| 8 | | | |
| 9 | | | |
| 10 | | | |
| 11 | | | |
| 12 | | | |
| 13 | | | |
| 14 | | | |
| 15 | | | |
| 16 | | | |
| 17 | | | |
| 18 | | | |
| 19 | | | |
| 20 | | | |
| 21 | | | |
| 22 | | | |
| 23 | | | |
| 24 | | | |
| 25 | | | |
| 26 | | | |
| 27 | | | |
| 28 | | | |
| 29 | | | |
| 30 | | | |
| 31 | | | |
| 32 | | | |
| 33 | | | |
| 34 | | | |
| 35 | | | |
| 36 | | | |
| 37 | | | |
| 38 | | | |
| 39 | | | |
| 40 | | | |
| 41 | | | |
| 42 | | | |
| 43 | | | |
| 44 | | | |
| 45 | | | |
| 46 | | | |
| 47 | | | |
| 48 | | | |
| 49 | | | |
| 50 | | | |
| 51 | | | |
| 52 | | | |
| 53 | | | |
| 54 | | | |
| 55 | | | |
| 56 | | | |
| 57 | | | |
| 58 | | | |
| 59 | | | |
| 60 | | | |
| 61 | | | |
| 62 | | | |
| 63 | | | |
| 64 | | | |
| 65 | | | |
| 66 | | | |
| 67 | | | |
| 68 | | | |
| 69 | | | |
| 70 | | | |
| 71 | | | |
| 72 | | | |
| 73 | | | |
| 74 | | | |
| 75 | | | |
| 76 | | | |
| 77 | | | |
| 78 | | | |
| 79 | | | |
| 80 | | | |
| 81 | | | |
| 82 | | | |
| 83 | | | |
| 84 | | | |
| 85 | | | |
| 86 | | | |
| 87 | | | |
| 88 | | | |
| 89 | | | |
| 90 | | | |
| 91 | | | |
| 92 | | | |
| 93 | | | |
| 94 | | | |
| 95 | | | |
| 96 | | | |
| 97 | | | |
| 98 | | | |
| 99 | | | |
| 100 | | | |

4.5 Experiment IV: Evaluation of Para-REGAL

4.5.1 Motivation and Objective.

4.5.2 Methodology.

Islands 1 & 2 *loose* *Island 3* *tight* *Island 0*

4.5.3 Para-REGAL Experiments.

$$\{ \dots, \dots, \dots, \dots, \dots \} = P_m - P_{cm} + \dots + \dots + \dots + \dots$$

4.6 Summary

V. Results and Analysis

the results of the experiments. In the following sections, we will first introduce the parallel hybrid GA and then present the experimental results.

5.1 Experiment I: Parallel Hybrid GA

In this section, we will introduce the parallel hybrid GA and its implementation. We will also present the experimental results of this experiment.

The parallel hybrid GA is a hybrid of a genetic algorithm and a parallel search algorithm. It uses a population-based search space and a parallel search space. The parallel search space is divided into several subspaces, each of which is assigned to a different processor. The processors communicate with each other through a message-passing interface.

5.1.1 Effectiveness Analysis.

In this section, we will analyze the effectiveness of the parallel hybrid GA. We will compare the performance of the parallel hybrid GA with the sequential hybrid GA and the parallel search algorithm.

We conducted several experiments to evaluate the performance of the parallel hybrid GA. The results show that the parallel hybrid GA is significantly faster than the sequential hybrid GA and the parallel search algorithm.

The parallel hybrid GA is able to find better solutions faster than the sequential hybrid GA and the parallel search algorithm. This is because the parallel hybrid GA is able to explore more of the search space simultaneously.

The parallel hybrid GA is also able to find better solutions faster than the sequential hybrid GA and the parallel search algorithm. This is because the parallel hybrid GA is able to explore more of the search space simultaneously.

The parallel hybrid GA is also able to find better solutions faster than the sequential hybrid GA and the parallel search algorithm. This is because the parallel hybrid GA is able to explore more of the search space simultaneously.

The parallel hybrid GA is also able to find better solutions faster than the sequential hybrid GA and the parallel search algorithm. This is because the parallel hybrid GA is able to explore more of the search space simultaneously.

The parallel hybrid GA is also able to find better solutions faster than the sequential hybrid GA and the parallel search algorithm. This is because the parallel hybrid GA is able to explore more of the search space simultaneously.

The parallel hybrid GA is also able to find better solutions faster than the sequential hybrid GA and the parallel search algorithm. This is because the parallel hybrid GA is able to explore more of the search space simultaneously.

The parallel hybrid GA is also able to find better solutions faster than the sequential hybrid GA and the parallel search algorithm. This is because the parallel hybrid GA is able to explore more of the search space simultaneously.

The parallel hybrid GA is also able to find better solutions faster than the sequential hybrid GA and the parallel search algorithm. This is because the parallel hybrid GA is able to explore more of the search space simultaneously.

The parallel hybrid GA is also able to find better solutions faster than the sequential hybrid GA and the parallel search algorithm. This is because the parallel hybrid GA is able to explore more of the search space simultaneously.

The parallel hybrid GA is also able to find better solutions faster than the sequential hybrid GA and the parallel search algorithm. This is because the parallel hybrid GA is able to explore more of the search space simultaneously.

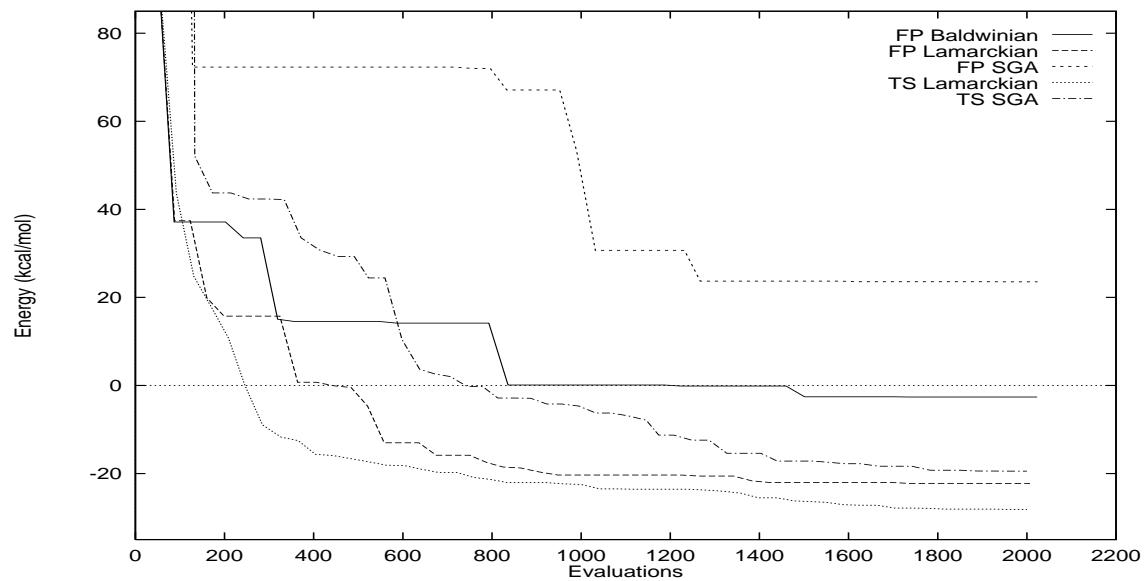
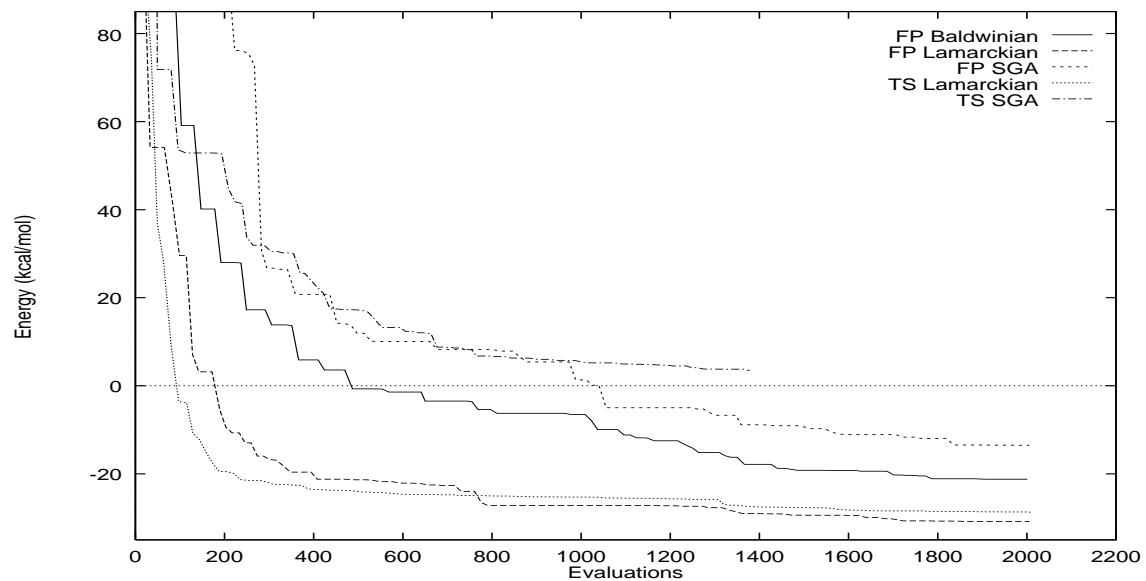
The parallel hybrid GA is also able to find better solutions faster than the sequential hybrid GA and the parallel search algorithm. This is because the parallel hybrid GA is able to explore more of the search space simultaneously.

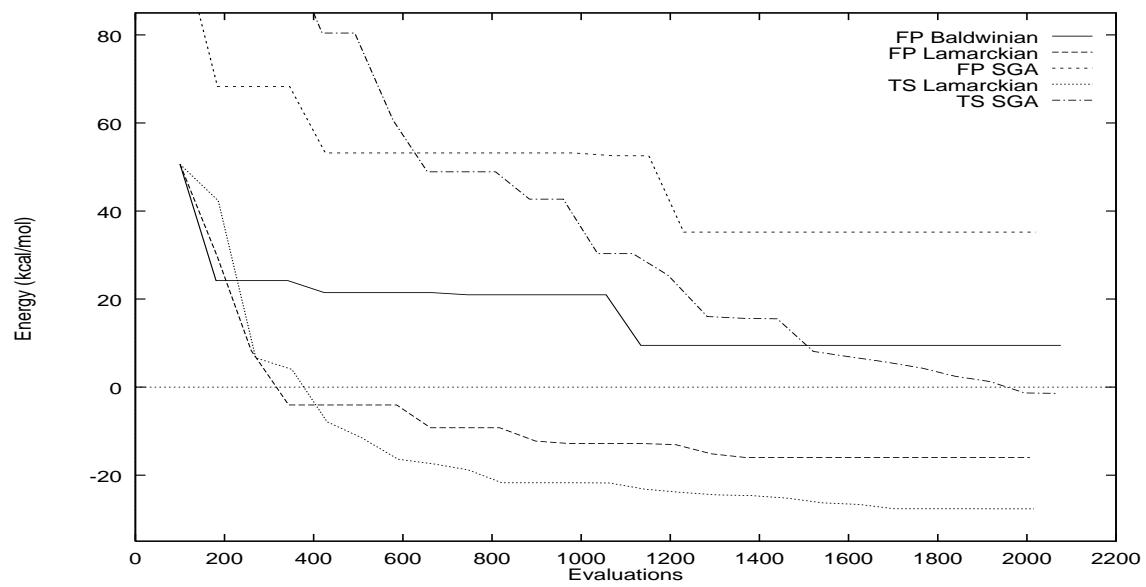
The parallel hybrid GA is also able to find better solutions faster than the sequential hybrid GA and the parallel search algorithm. This is because the parallel hybrid GA is able to explore more of the search space simultaneously.

The parallel hybrid GA is also able to find better solutions faster than the sequential hybrid GA and the parallel search algorithm. This is because the parallel hybrid GA is able to explore more of the search space simultaneously.

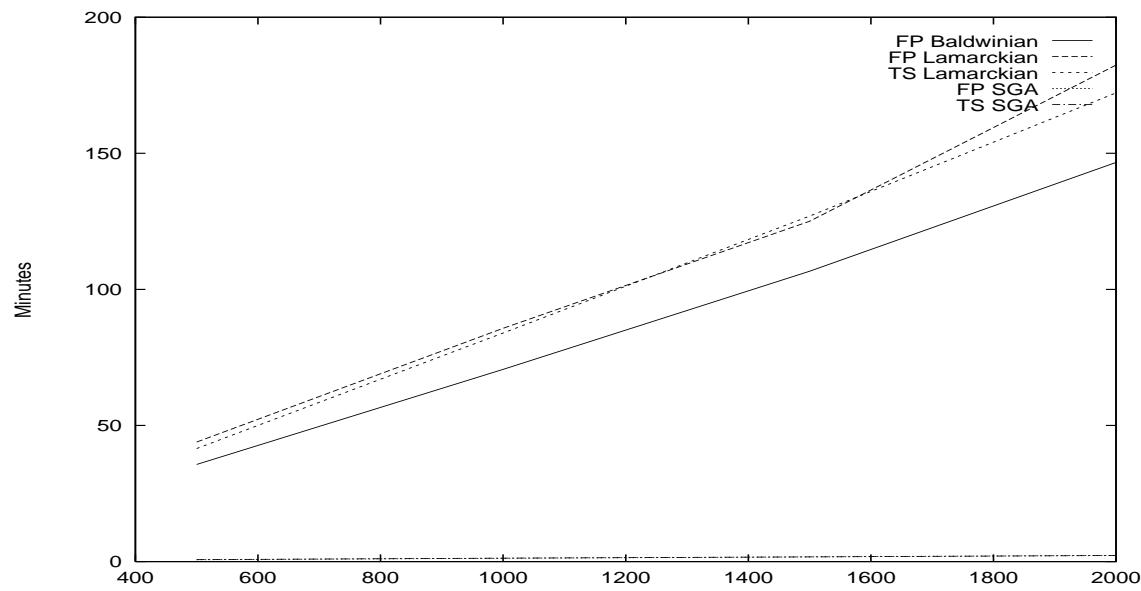
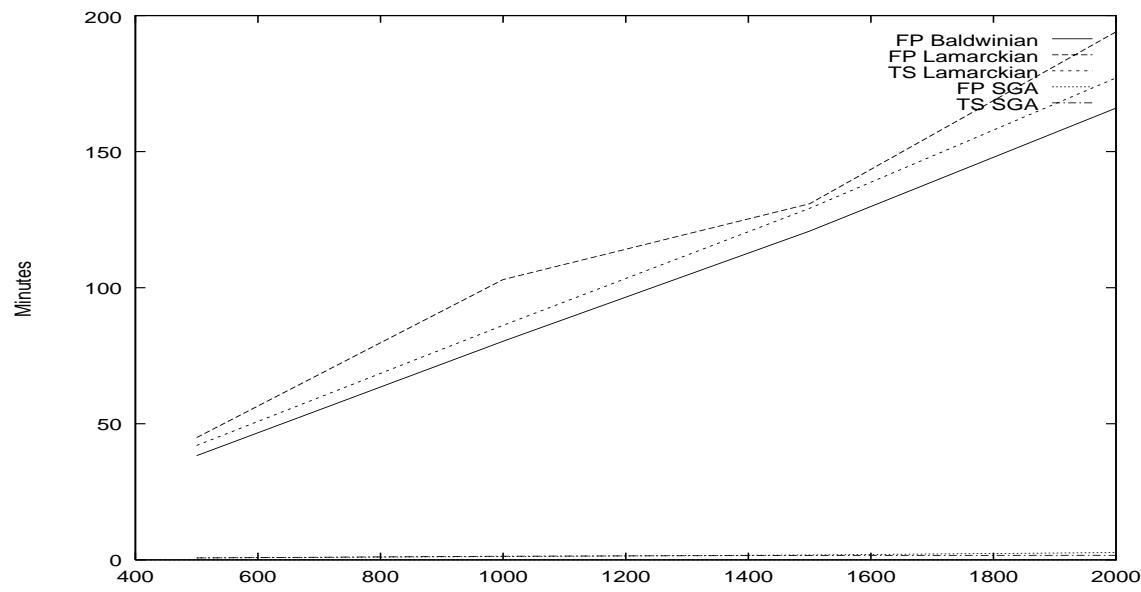
The parallel hybrid GA is also able to find better solutions faster than the sequential hybrid GA and the parallel search algorithm. This is because the parallel hybrid GA is able to explore more of the search space simultaneously.

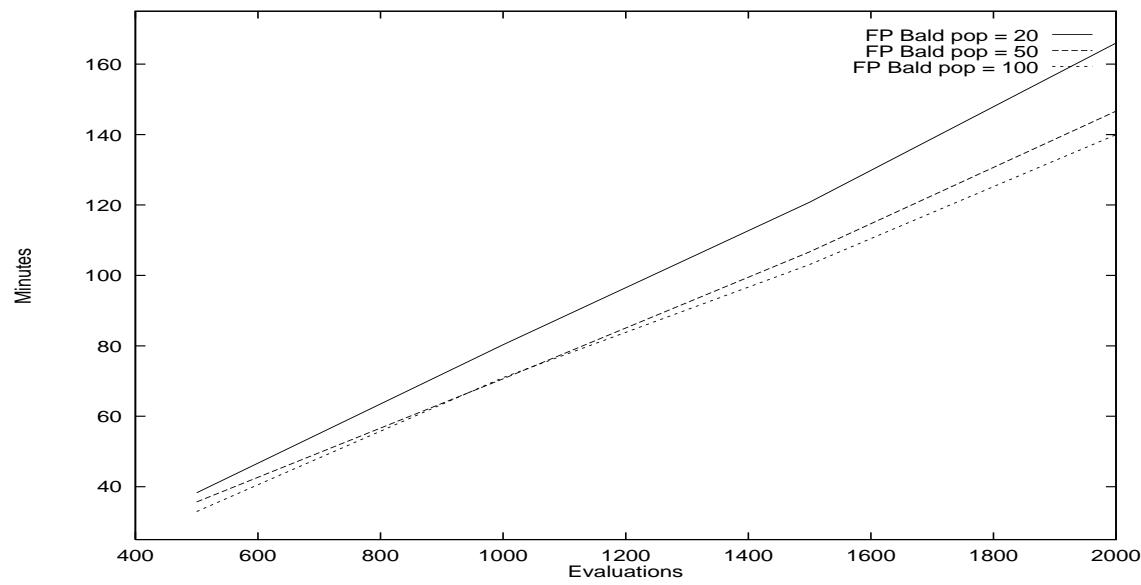
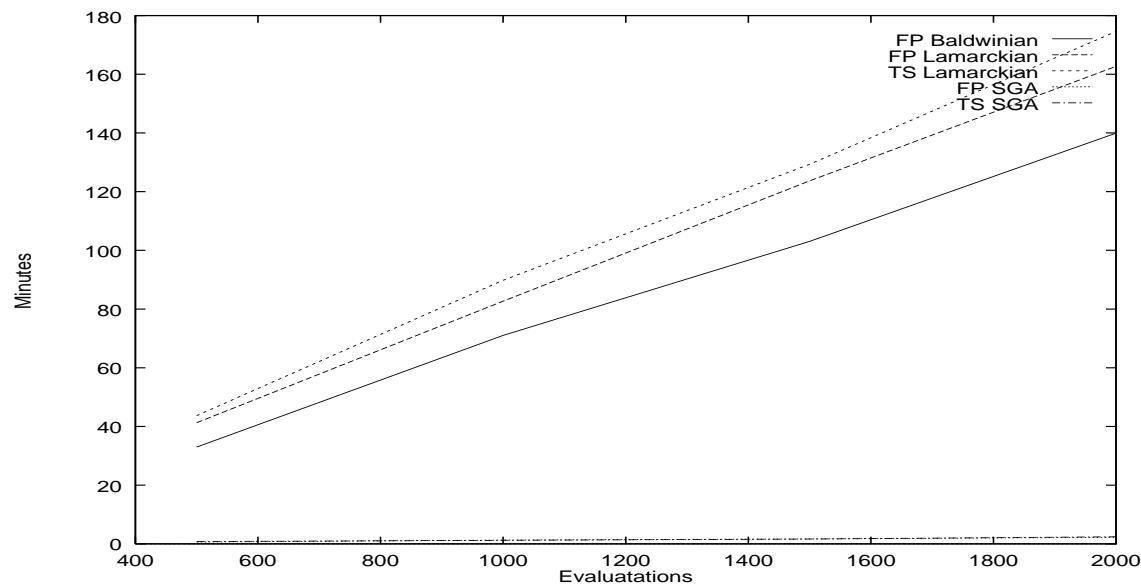
The parallel hybrid GA is also able to find better solutions faster than the sequential hybrid GA and the parallel search algorithm. This is because the parallel hybrid GA is able to explore more of the search space simultaneously.

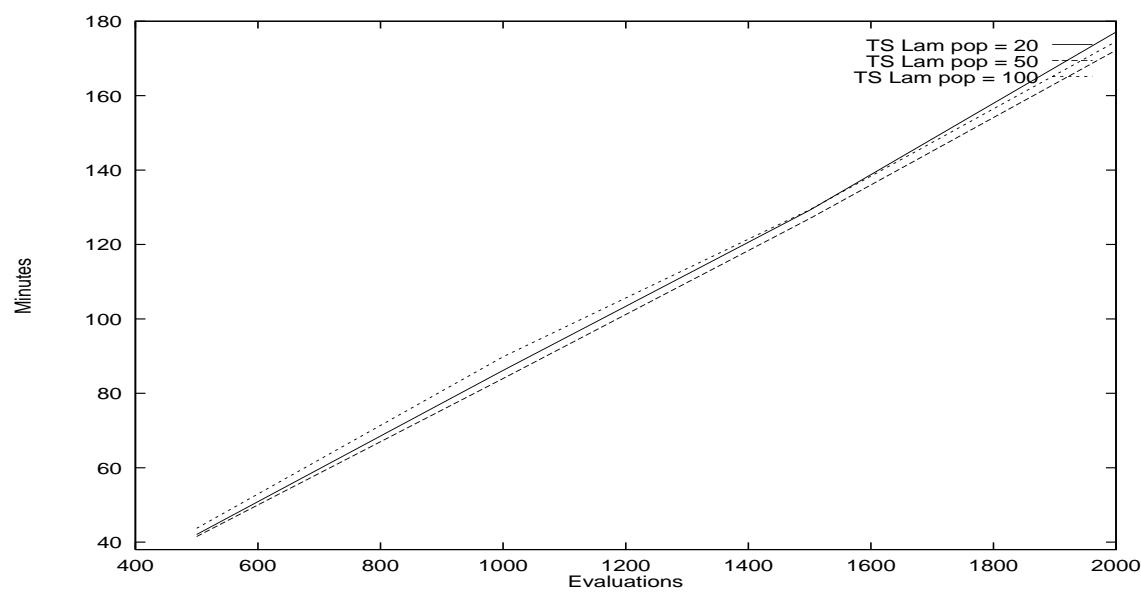
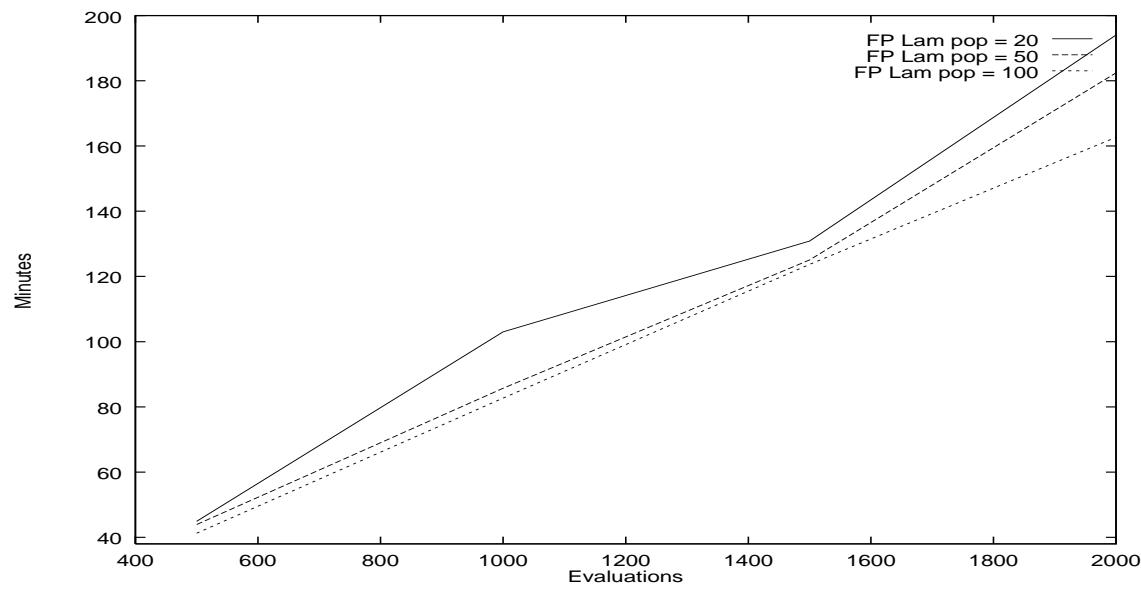


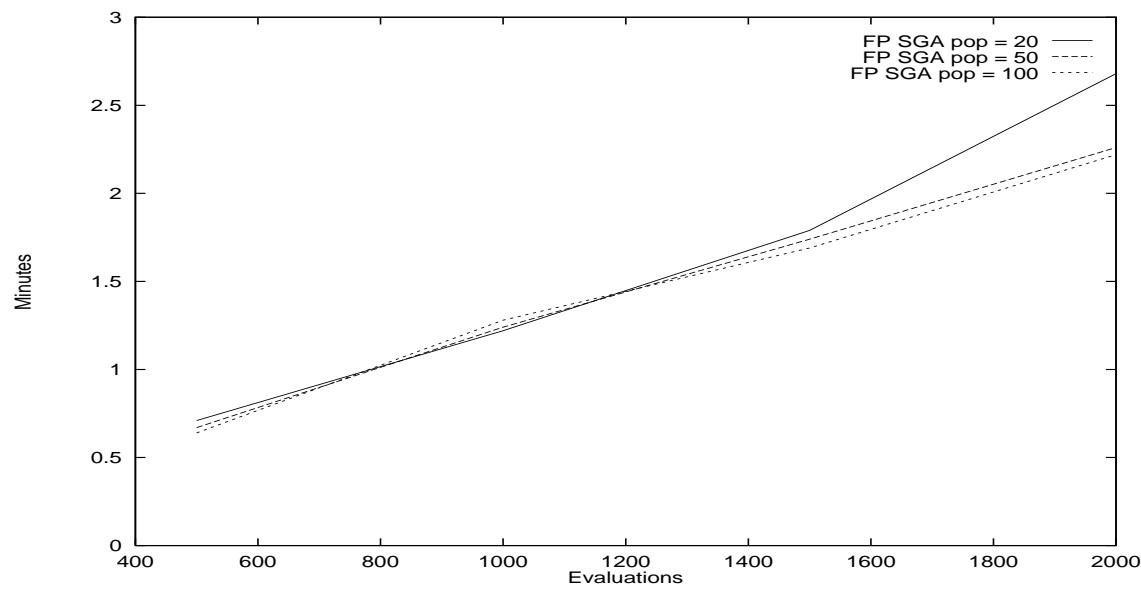
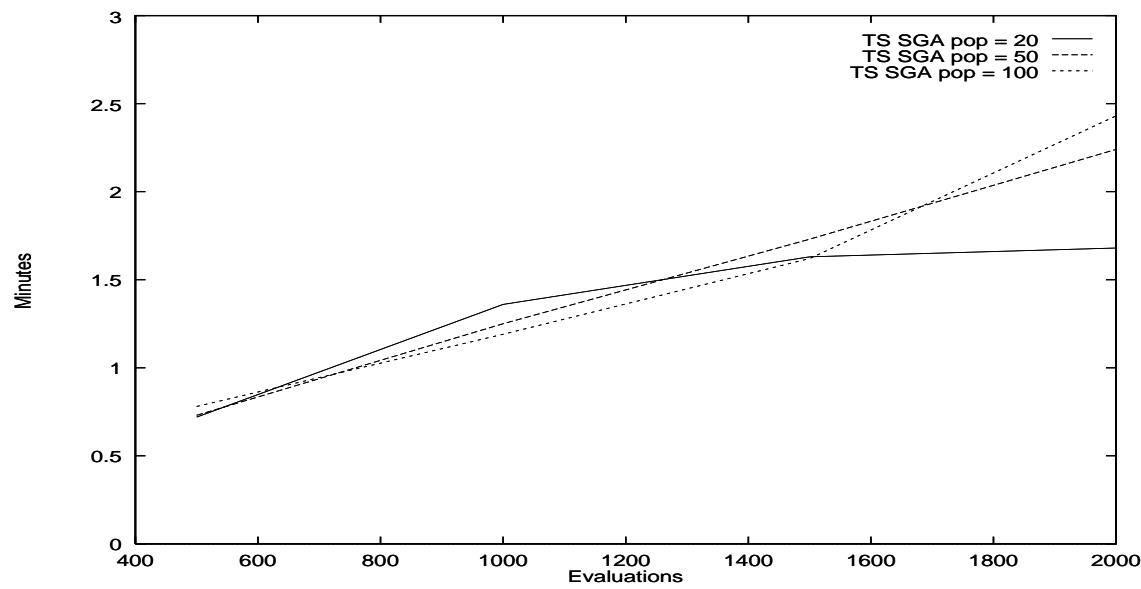


5.1.2 Efficiency Analysis, Serial.







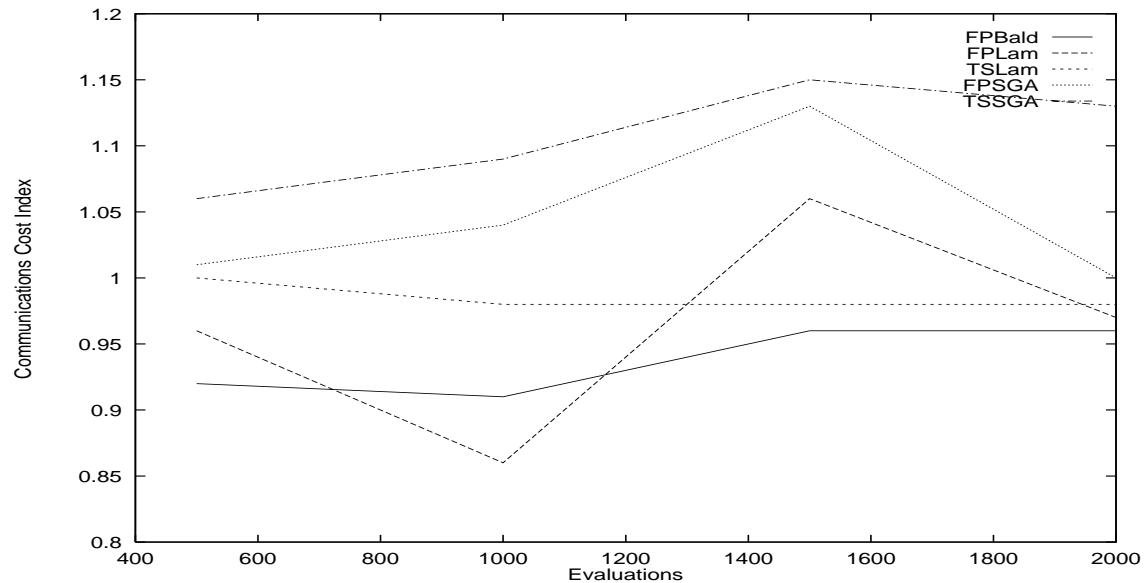


5.1.3 Efficiency Analysis, Parallel.

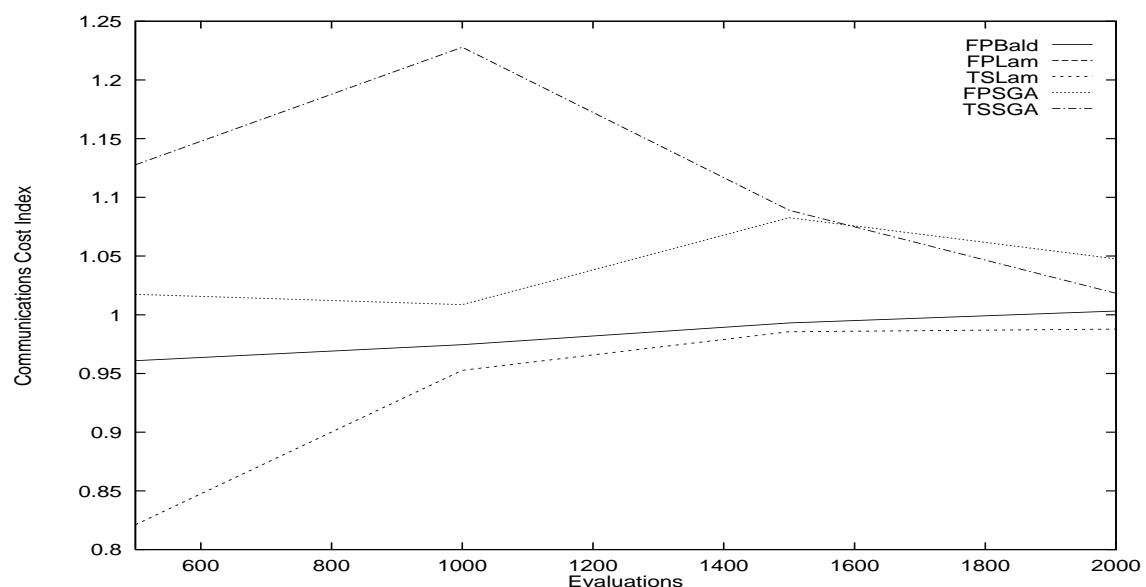
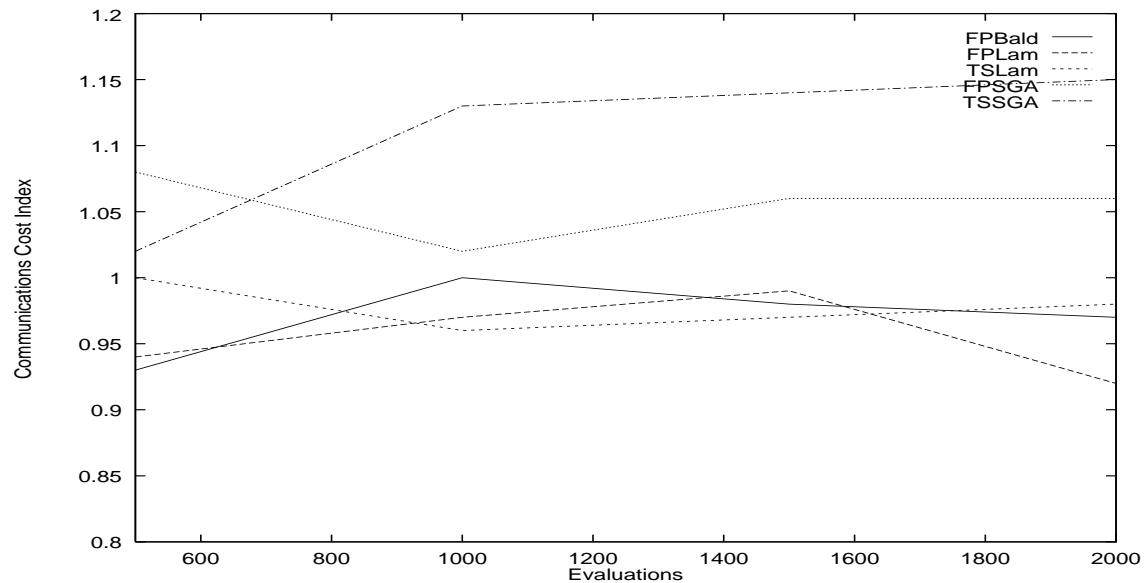
$$C_{communications} = T_{P_2} - T_S$$

$$C_{communications} = \frac{T_{P_2}}{T_S}$$

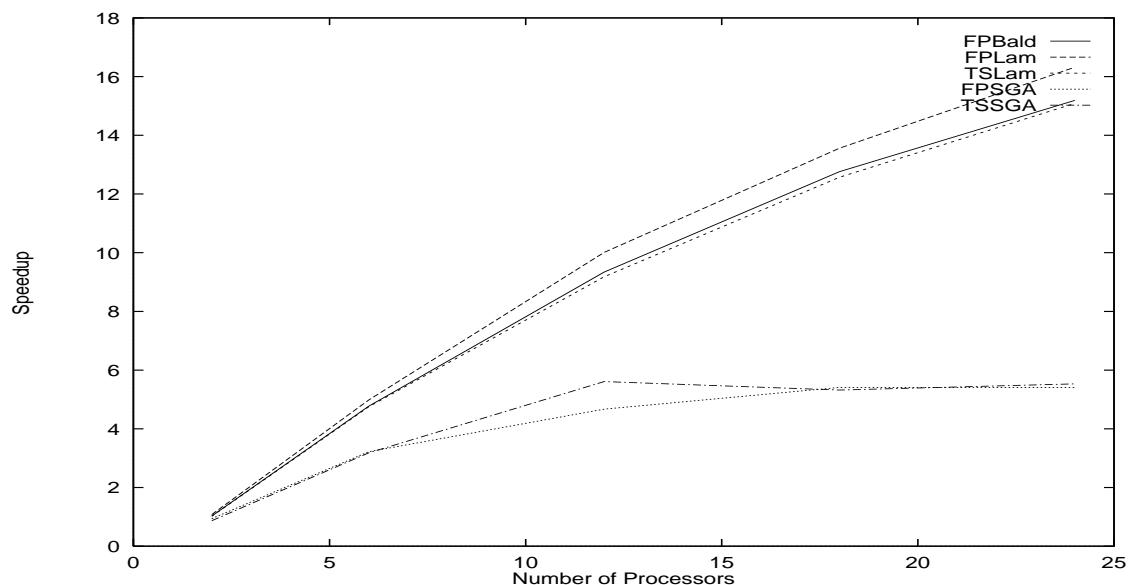
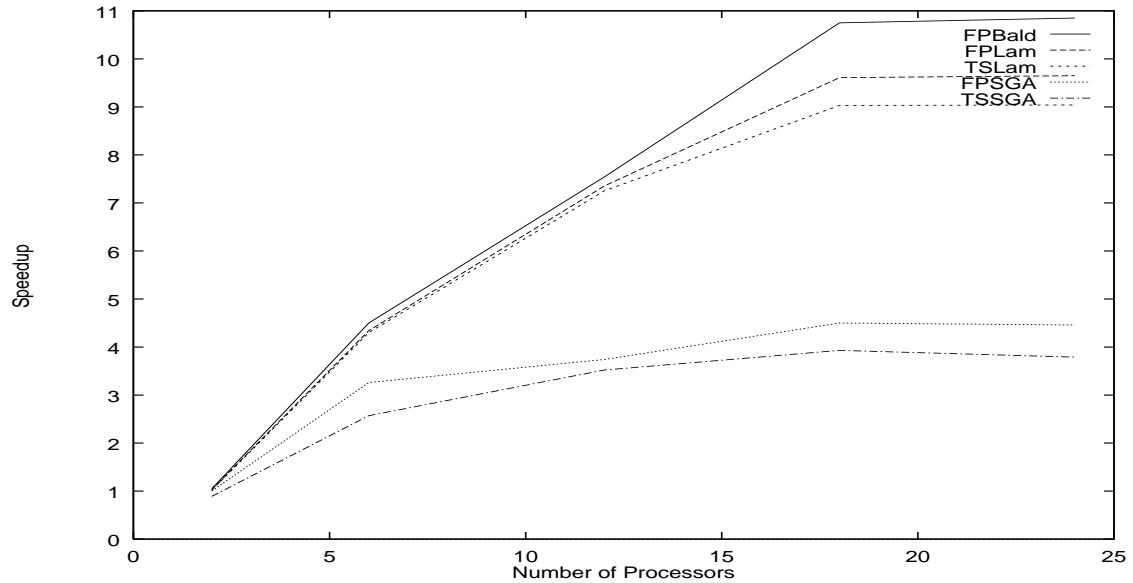
$$\text{Communication Cost Index} = \frac{T_{P_2}}{T_S}$$

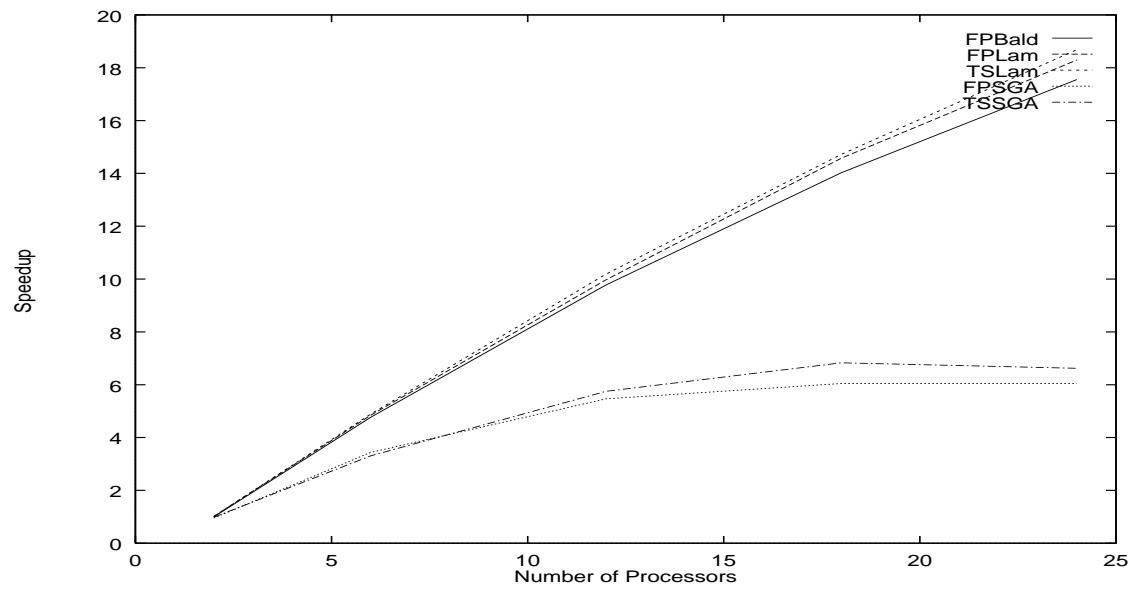


$$S = \frac{T_S}{T_P}$$



S T_S $best$ T_P
 p $S > p$ $linear\ speedup\ S$ T_S
 \dots $super\ linear\ speedup$ \dots T_P



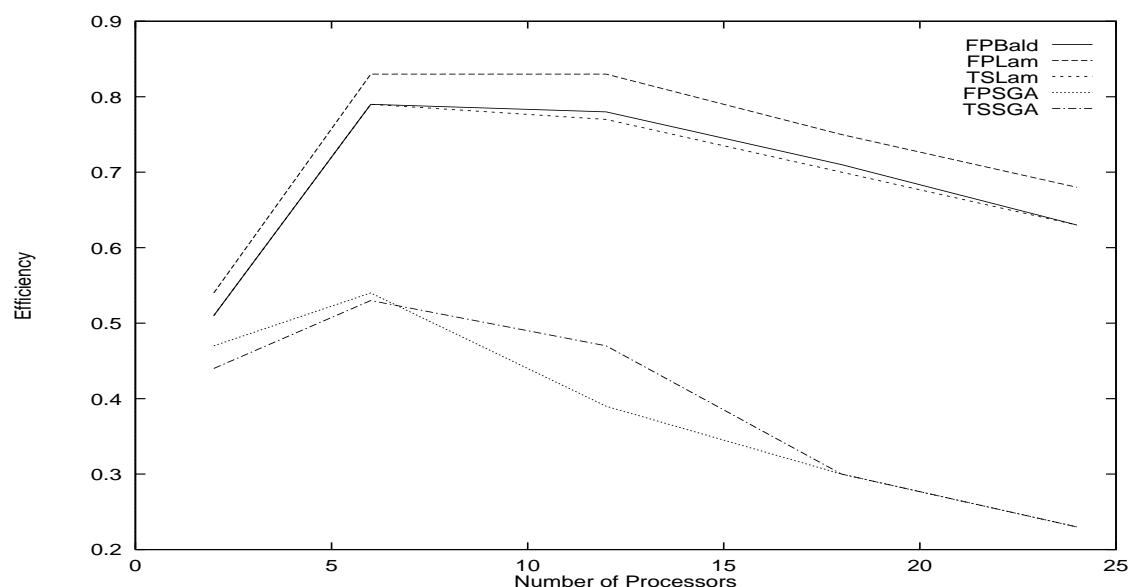
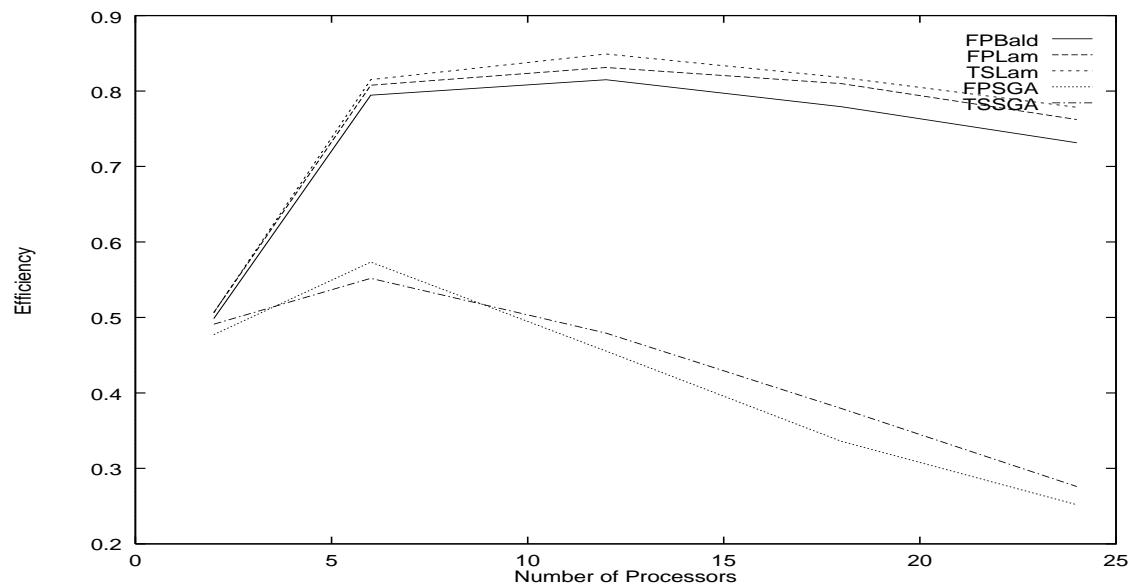


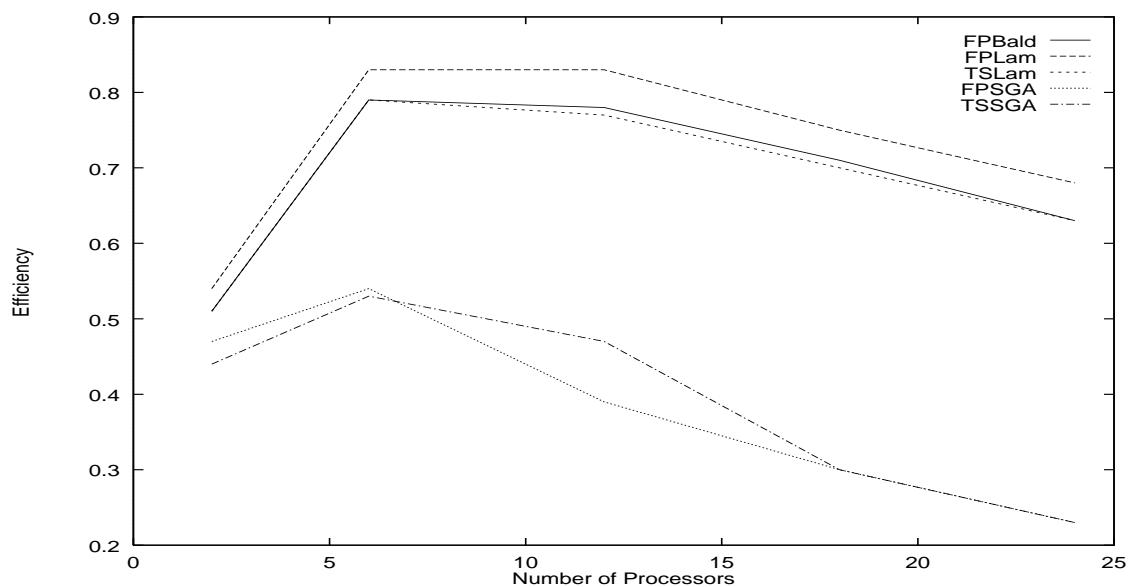
the cost-optimal solution. In fact, we can prove that the cost-optimal solution is the one that minimizes the ratio $\frac{S}{p}$. This is shown in the following theorem.

Theorem 5.1 $E(S, p)$ is cost-optimal if and only if $E(S, p) = \min_{E(S, p') \in E} \frac{S}{p'}$.

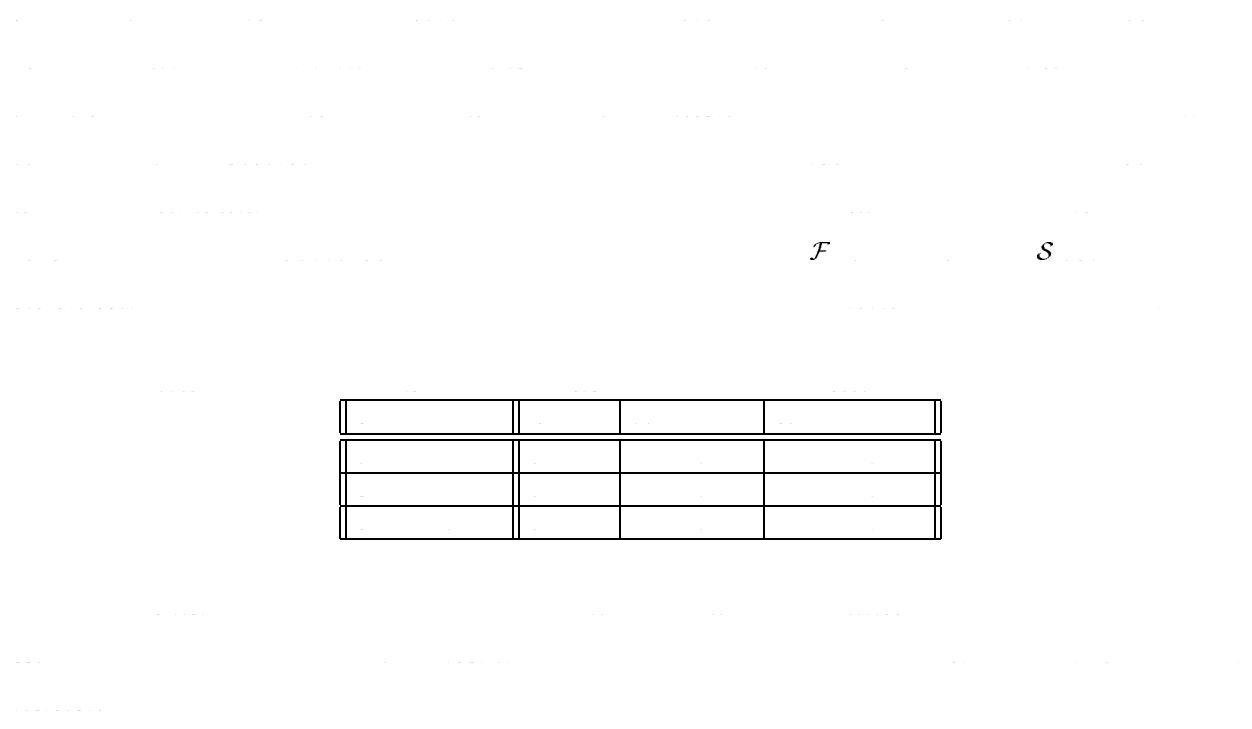
Proof We first prove that if $E(S, p)$ is cost-optimal, then $E(S, p) = \min_{E(S, p') \in E} \frac{S}{p'}$. Let E^* be the cost-optimal solution. Then, by definition, E^* is the solution that minimizes the cost function $Cost(E)$. We will show that E^* also minimizes the ratio $\frac{S}{p}$. To do this, we will show that any other solution E' that has a lower cost than E^* must have a higher ratio $\frac{S}{p}$ than E^* . Suppose, for contradiction, that E' has a lower cost than E^* but a lower ratio $\frac{S}{p}$ than E^* . Then, we can write $E' = (S', p')$ where $S' < S$ and $p' < p$. Since E' has a lower cost than E^* , we have $Cost(E') < Cost(E^*)$. But since E^* is cost-optimal, we have $Cost(E^*) \leq Cost(E')$. This contradicts our assumption that E' has a lower cost than E^* . Therefore, E^* must minimize the ratio $\frac{S}{p}$.

5.2 Experiment II: Preliminary REGAL Evaluation





5.2.1 *[Met]-enkephalin.*



| | | | |
|--|--|--|--|
| | | | |
| | | | |
| | | | |
| | | | |
| | | | |
| | | | |
| | | | |

5.2.2 Polyalanine.

step

α

α

| | | | |
|--|--|--|--|
| | | | |
| | | | |
| | | | |
| | | | |
| | | | |
| | | | |
| | | | |

5.2.3 Efficiency.

| | | | 1 |
|--|--|--|---|
| | | | |
| | | | |

5.3 Experiment III: Analysis of Exogenous Parameters for REGAL

independent

• *current best trajectory* \rightarrow *current best trajectory* \rightarrow *current best trajectory*

• *current best trajectory* \rightarrow *current best trajectory* \rightarrow *current best trajectory*

2

• *current best trajectory* \rightarrow *current best trajectory* \rightarrow *current best trajectory*

3

²Analysis of Variance, see Appendix F.1 for more details. Concern has been raised about lack of variability because a single seed set was used. The Kruskal-Wallis H Test (Appendix F.2) was used as an independent method to verify the ANOVA results. The conclusions were the same. Kruskal-Wallis results are not shown.

³Hypothesis testing was not done on run times because system loading in the multi-user environment could not be controlled. They are provided for reference only. However, the large number of experiments tends to dampen out cases where the platform was heavily loaded. Thus, the data are insightful.

| | | | | | | |
|--|--|--|--|--|--|--|
| | | | | | | |
| | | | | | | |
| | | | | | | |
| | | | | | | |
| | | | | | | |
| | | | | | | |
| | | | | | | |
| | | | | | | |
| | | | | | | |
| | | | | | | |
| | | | | | | |
| | | | | | | |
| | | | | | | |
| | | | | | | |
| | | | | | | |
| | | | | | | |

⁴This nomenclature is from the GENOCOP-III documentation. It would be more accurate to stay the reference population is operated upon.

| F_0 | α | $*$ |
|-------|----------|-----|
| | | |
| | | |
| | | |
| | | |
| | | * |
| | | * |
| | | |
| | | |
| | | |
| | | |
| | | |
| | | |
| | | |
| | | |
| | | |
| | | |
| | | |
| | | |
| | | |
| | | |

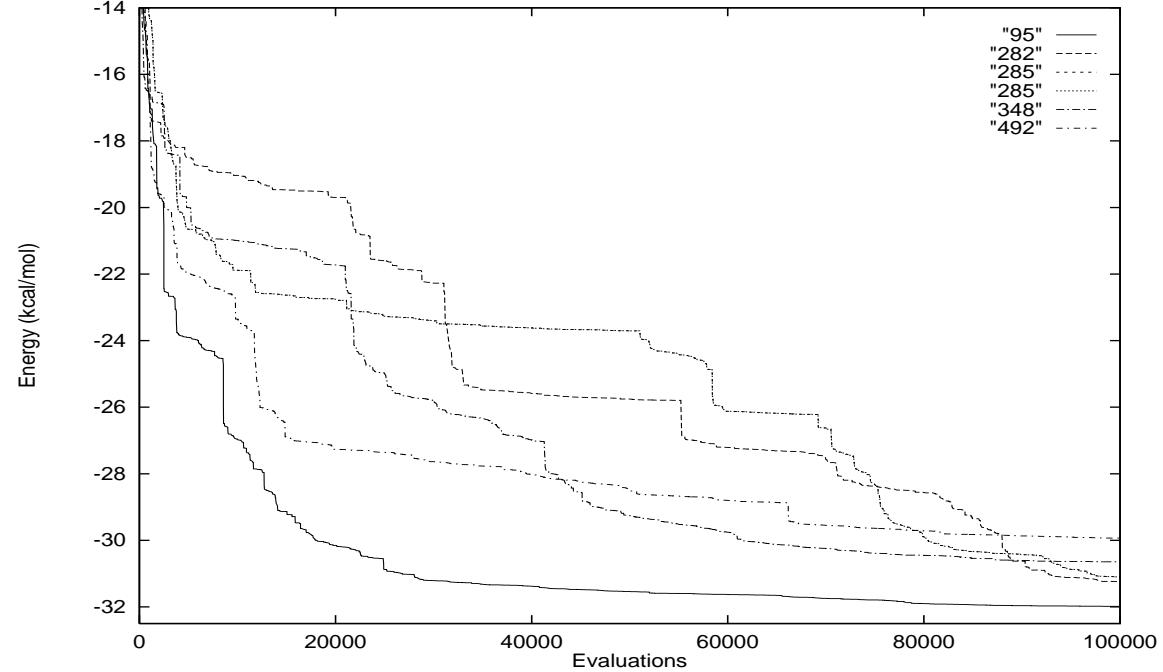
| | | |
|----------|-------|----------|
| α | H_0 | α |
| | | |
| | | |
| | | |

| | | |
|----------|----------|----------|
| α | α | α |
| | | |
| | | |
| | | |

5.4 Experiment IV: Analysis of Para-REGAL

Island 3

| | | | | | | | | | | | |
|--|--|--|--|--|--|--|--|--|--|--|--|
| | | | | | | | | | | | |
| | | | | | | | | | | | |
| | | | | | | | | | | | |
| | | | | | | | | | | | |
| | | | | | | | | | | | |
| | | | | | | | | | | | |
| | | | | | | | | | | | |
| | | | | | | | | | | | |



periodicity

$$\begin{aligned}
 & n = \frac{m}{n} \quad m = \frac{n}{m} \\
 & -3 \quad th \\
 & \vdots \\
 & = K
 \end{aligned}$$

| | P_m | P_{cm} | | | |
|--|-------|----------|--|--|--|
| | | | | | |
| | | | | | |
| | | | | | |
| | | | | | |
| | | | | | |
| | | | | | |
| | | | | | |
| | | | | | |
| | | | | | |
| | | | | | |
| | | | | | |
| | | | | | |
| | | | | | |
| | | | | | |
| | | | | | |
| | | | | | |
| | | | | | |
| | | | | | |

5.5 Summary

⁵Considerable resources is a relative concept. Even a 1000 hours of computer time is trivial when compared to experimental techniques that require years to yield results.

VI. Conclusions and Recommendation

The results of this investigation have shown that the proposed system can be used to detect and correct errors in the design of aircraft structures. The system has been able to identify potential problems in the design of the aircraft structure and provide recommendations for their correction. The system has also been able to identify potential problems in the design of the aircraft structure and provide recommendations for their correction. The system has also been able to identify potential problems in the design of the aircraft structure and provide recommendations for their correction.

Grand Challenge

6.1 Initiative I: PHGA

The first initiative, PHGA, is designed to help designers identify potential problems in the design of aircraft structures. The system uses a genetic algorithm to search for solutions to a set of constraints. The system has been able to identify potential problems in the design of the aircraft structure and provide recommendations for their correction. The system has also been able to identify potential problems in the design of the aircraft structure and provide recommendations for their correction. The system has also been able to identify potential problems in the design of the aircraft structure and provide recommendations for their correction.

6.2 Initiative II: REGAL

The second initiative, REGAL, is designed to help designers identify potential problems in the design of aircraft structures. The system uses a genetic algorithm to search for solutions to a set of constraints. The system has been able to identify potential problems in the design of the aircraft structure and provide recommendations for their correction. The system has also been able to identify potential problems in the design of the aircraft structure and provide recommendations for their correction. The system has also been able to identify potential problems in the design of the aircraft structure and provide recommendations for their correction.

¹ Actual implementation is out of scope for this investigation because research is required into appropriate control metrics.

² Limit the dihedral angle's range to a lower bound greater than $-\pi$ and an upper bound less than π .

fundamental theorem of genetic algorithms

the fundamental theorem of genetic algorithms (FTGA) is a well-known result in the field of genetic algorithms (GAs). It states that a GA can search a large space of solutions and find the global optimum with probability one if certain conditions are met. These conditions include a sufficiently large population size, a high mutation rate, and a low crossover rate.

6.3 Initiative III: Examination of Exogenous Parameters

the examination of exogenous parameters is a key component of Initiative III. This initiative aims to identify and analyze the external factors that influence the performance of GAs. These factors include the problem domain, the search space, and the environment in which the GA operates.

the examination of exogenous parameters involves several steps. First, the problem domain is identified and characterized. This includes determining the type of optimization problem (e.g., linear or nonlinear), the constraints, and the objective function.

second, the search space is analyzed. this involves identifying the dimensions of the search space and determining the range of values for each dimension. this information is used to guide the search process and ensure that it explores the entire search space effectively.

third, the environment in which the GA operates is examined. this includes identifying the external factors that may affect the performance of the GA, such as temperature, humidity, and noise levels.

6.4 Initiative IV: Para-REGAL

the para-regal initiative is a new approach to solving optimization problems. it combines the strengths of traditional GAs and parallel computing to achieve faster convergence and better performance.

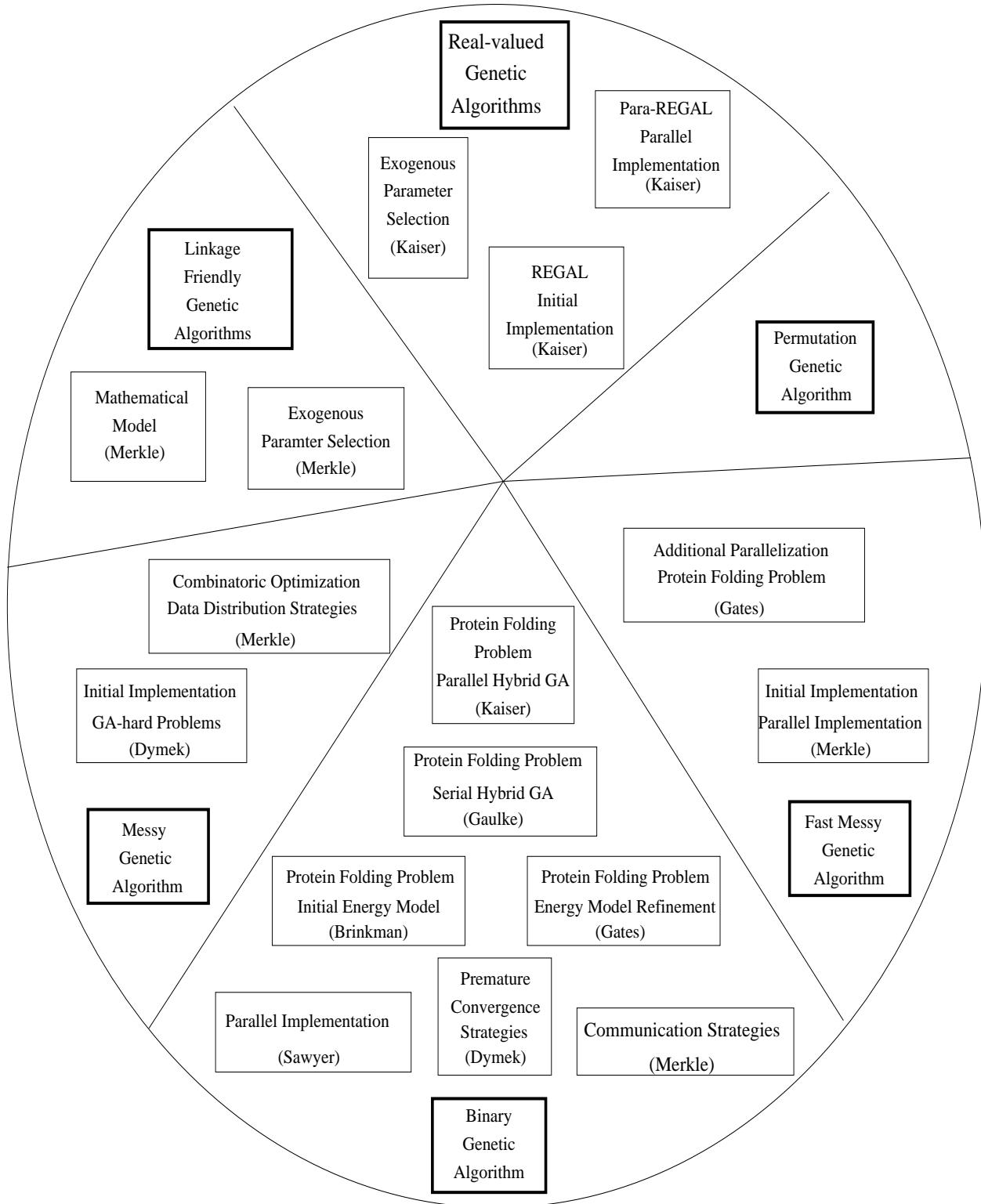
the para-regal initiative involves several key components. first, it uses a parallel search space to explore multiple regions of the search space simultaneously. this allows the algorithm to converge faster and find better solutions.

second, the para-regal initiative uses a hybrid crossover operator that combines the strengths of different crossover operators. this allows the algorithm to explore different regions of the search space more effectively and find better solutions.

third, the para-regal initiative uses a self-adaptive mutation operator that adapts the mutation rate based on the current state of the search process. this allows the algorithm to explore different regions of the search space more effectively and find better solutions.

the para-regal initiative has shown promising results in solving complex optimization problems. it has been successfully applied to a variety of problems, including function optimization, feature selection, and classification.

the para-regal initiative is a significant advancement in the field of optimization. it has the potential to revolutionize the way we solve optimization problems and open up new opportunities for applications in various fields.



6.5 Recommendations

- The first recommendation is to increase the number of students in the program. This will help to reduce the workload of the current students and faculty members.
- The second recommendation is to improve the facilities available to the students. This includes better computer labs, more study areas, and improved transportation options.
- The third recommendation is to increase the number of faculty members. This will help to provide more individual attention to each student and ensure that they receive the best possible education.
- The fourth recommendation is to improve the curriculum. This includes adding new courses and updating existing ones to reflect the latest developments in the field.
- The fifth recommendation is to increase the number of internships available to the students. This will help them to gain practical experience and prepare for their future careers.
- The sixth recommendation is to improve the placement services offered by the university. This includes helping students find jobs and providing them with the necessary resources to succeed in their chosen fields.
- The seventh recommendation is to increase the number of research opportunities available to the students. This will help them to develop their skills and contribute to the field.
- The eighth recommendation is to improve the overall atmosphere of the program. This includes creating a supportive environment where students can feel comfortable asking questions and seeking help.
- The ninth recommendation is to increase the number of scholarships available to the students. This will help them to offset the cost of tuition and other expenses.
- The tenth recommendation is to improve the communication between the students and the faculty members. This will help to ensure that all parties are on the same page and working towards the same goals.

6.6 Summary

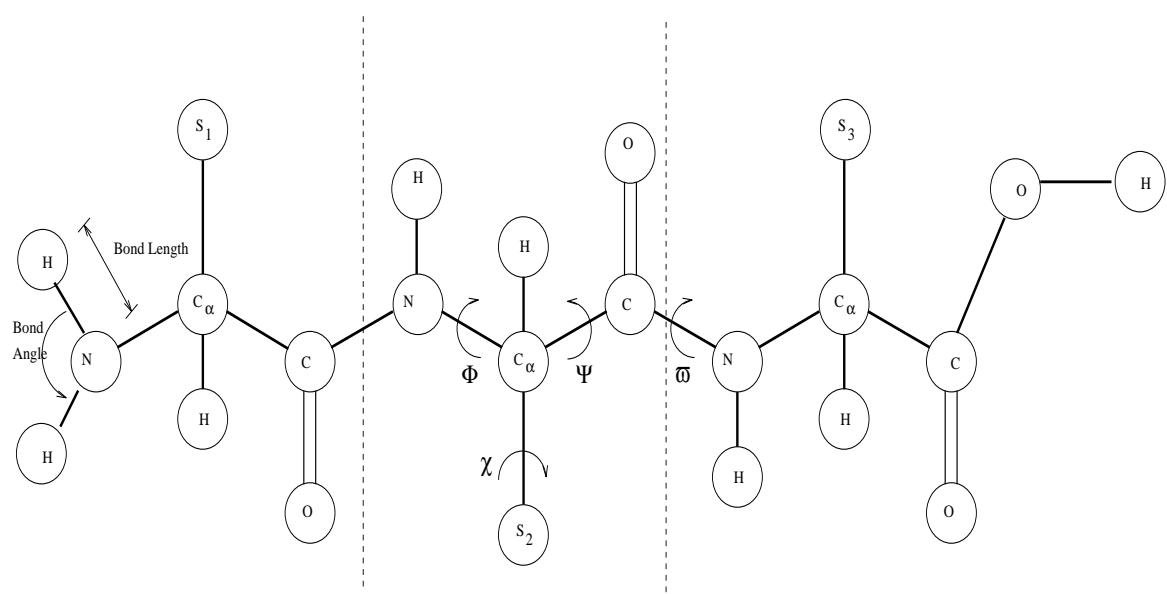
- The summary of the recommendations is as follows:
 - Increase the number of students in the program.
 - Improve the facilities available to the students.
 - Increase the number of faculty members.
 - Improve the curriculum.
 - Increase the number of internships available to the students.
 - Improve the placement services offered by the university.
 - Increase the number of research opportunities available to the students.
 - Improve the overall atmosphere of the program.
 - Increase the number of scholarships available to the students.
 - Improve the communication between the students and the faculty members.

Appendix A. Background on the Protein Folding and Protein Structure Prediction Problems

The protein folding problem is the problem of determining the three-dimensional structure of a protein from its primary sequence. It is also known as the inverse folding problem. The protein structure prediction problem is the problem of determining the three-dimensional structure of a protein from its primary sequence and some other information such as the secondary structure or the contact map. It is also known as the design problem.

A.1 Introduction to Proteins and Associated Terminology

A protein is a linear chain of amino acids. The backbone of a protein is the chain of carbon atoms that connect the amino acid residues. The side-chain of a protein is the part of the molecule that is attached to the backbone at each residue position. The primary structure of a protein is the sequence of amino acids in the backbone. The residues are the individual amino acid units that make up the protein. The α -carbon is the central carbon atom in the backbone of an amino acid residue. The β -carbon is the carbon atom that is bonded to the side-chain of an amino acid residue. The γ -carbon is the carbon atom that is bonded to the β -carbon of the previous residue in the backbone. The δ -carbon is the carbon atom that is bonded to the γ -carbon of the previous residue in the backbone.



β -sheets

secondary structure

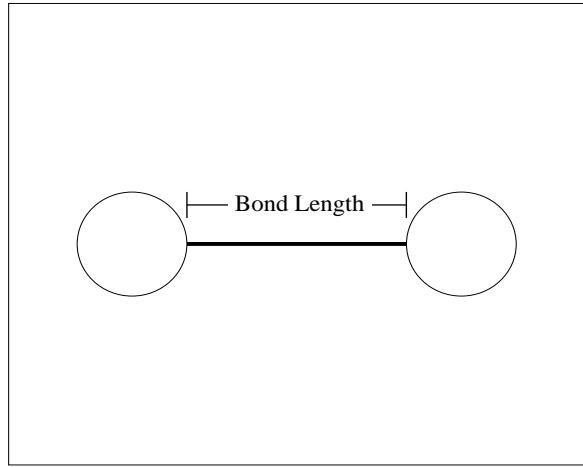
ϕ ψ

α -helices

ϕ, ψ

| ϕ, ψ | ϕ | ψ |
|--------------|--------|--------|
| α | | |
| α | | |
| 10 | | |
| β | | |
| β | | |
| | | |
| | | |
| | | |
| | | |
| | | |

tertiary structure conformation

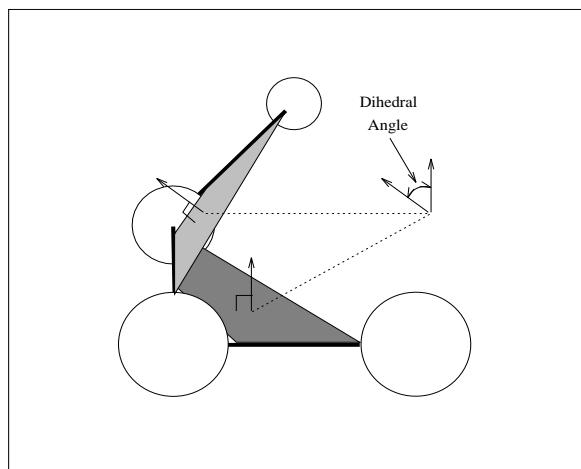
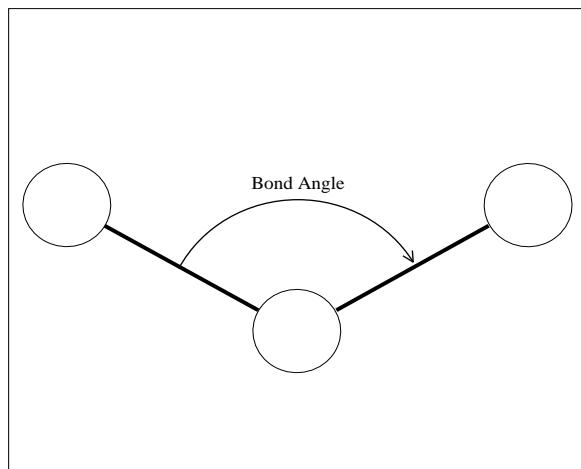


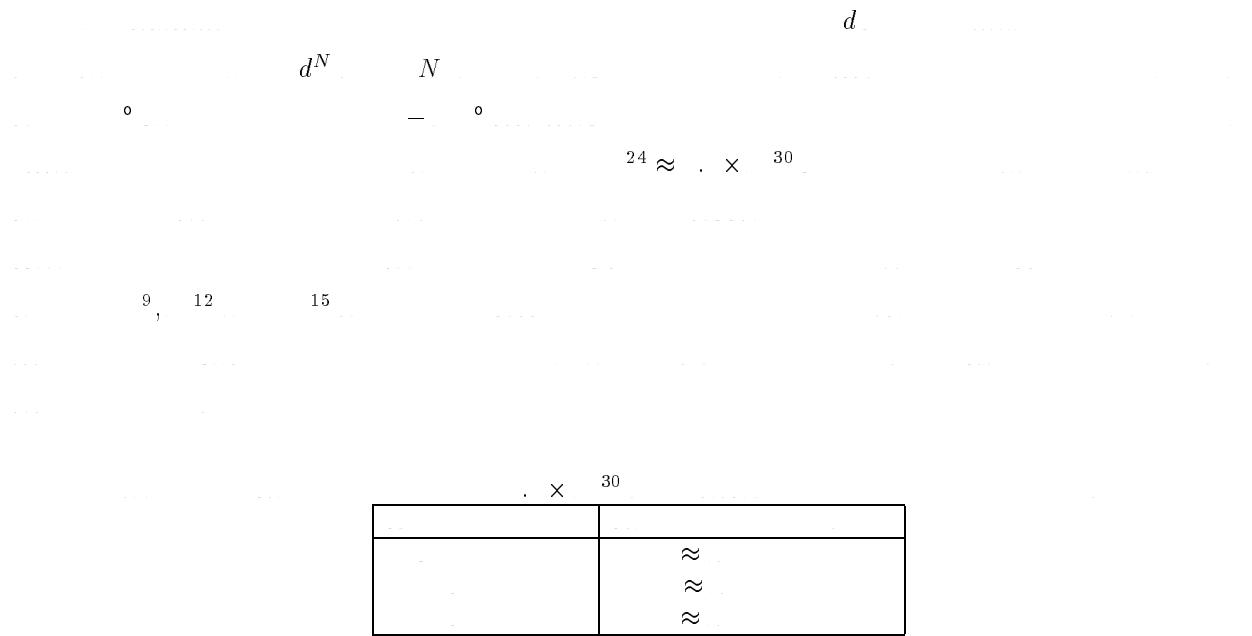
native state ($\phi = \psi = \omega = \chi_i = 0$)

bond length ($d_{ij} = \sqrt{(x_i - x_j)^2 + (y_i - y_j)^2 + (z_i - z_j)^2}$)

bond angle ($C_{\alpha_1} = \frac{\phi}{n}, C_{\alpha_2} = \frac{\psi}{n}, C_{\alpha_3} = \frac{\omega}{n}$)

dihedral angle (χ_i)





A.2 Experimental Tertiary Structure Determination

The experimental tertiary structure determination process involves several steps:

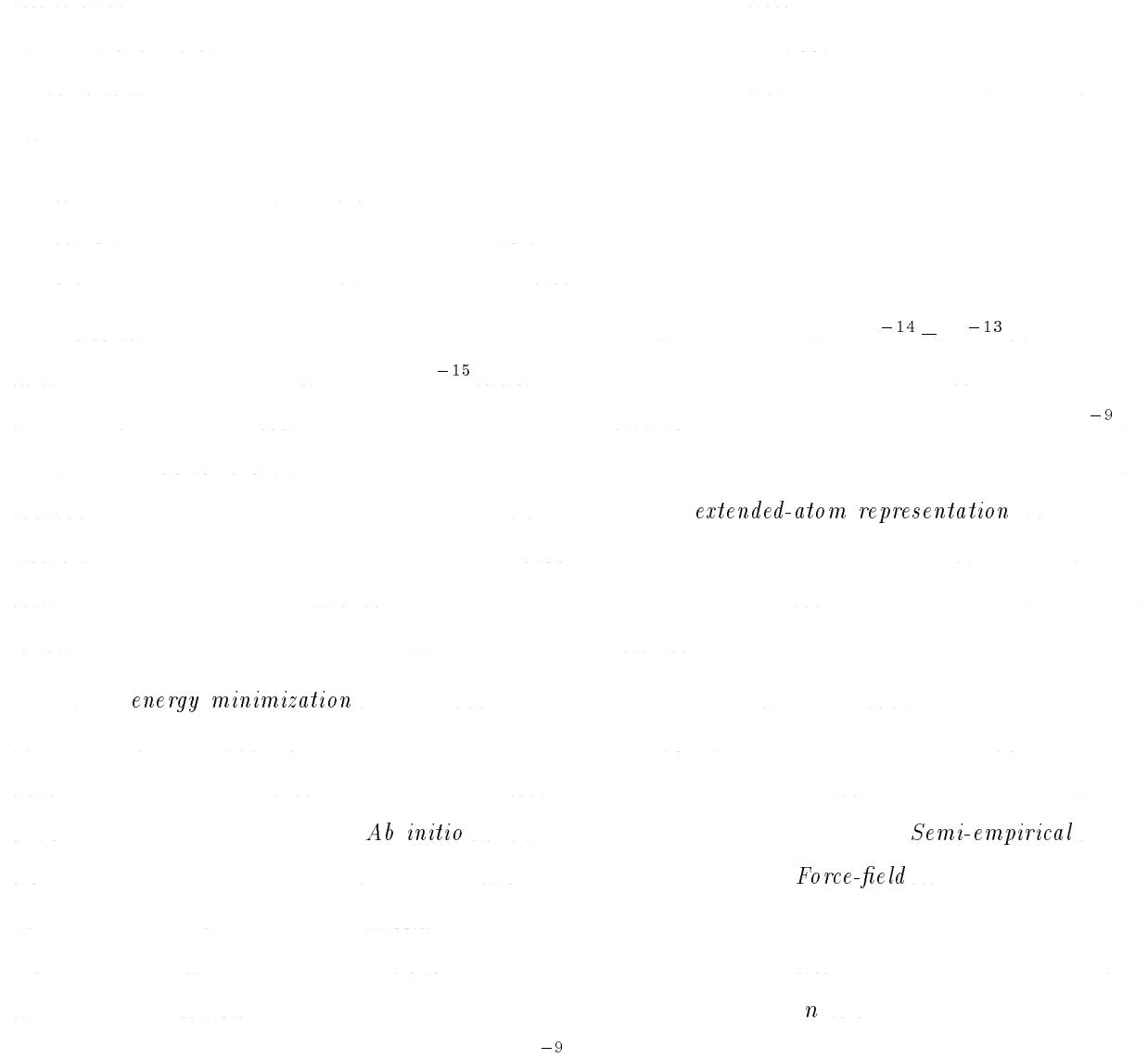
- Protein Data Bank (PDB) Search:** The first step is to search the PDB for existing structures. This is done using various search criteria such as sequence similarity, structural motifs, and chemical environment.
- Homology Modeling:** If no exact matches are found, homology modeling is used to predict the structure based on the closest found template. This involves aligning the target sequence with the template and using the template's structure as a starting point for the model.
- Refinement:** The predicted structure is refined using molecular dynamics simulations and energy minimization to improve its quality and fit to the experimental data.
- Validation:** The final structure is validated against experimental data, such as X-ray diffraction or NMR spectra, to ensure its accuracy and reliability.

A.3 Tertiary Structure Prediction (PFP)

The PFP process follows a similar workflow to experimental structure determination, but it uses computational methods instead of experimental data:

- Sequence Alignment:** The target sequence is aligned with a database of known protein structures to identify potential templates.
- Template Selection:** The best template is selected based on factors such as sequence identity, structural similarity, and conservation.
- Model Building:** A homology model is built by aligning the target sequence with the template's structure.
- Refinement:** The model is refined using computational techniques like molecular dynamics and energy minimization.
- Prediction Accuracy:** The predicted structure is evaluated against experimental data to determine its accuracy. The term "Exact" is often used to describe a prediction that perfectly matches the experimental structure.

A.3.1 Classical Prediction Methods. Molecular dynamics



| | | n |
|-----------------------|-------------------------------------|-----|
| <i>ab initio</i> | $\mathcal{O} n^5$ | |
| <i>semi-empirical</i> | $\mathcal{O} n^4 - \mathcal{O} n^3$ | |
| <i>force-field</i> | $\mathcal{O} n^2$ | |

$$E = \sum_{\substack{(i,j,k,l) \in \mathcal{D} \\ (i,j) \in \mathcal{N} \\ (i,j) \in \mathcal{H}}} \frac{U_{0_{ijkl}}}{n_{ijkl}} \pm n_{ijkl} \cdot \frac{\epsilon_{ij}}{F_{ij}} \cdot \frac{r_0}{r_{ij}}^{12} - \cdot \cdot \cdot \frac{\epsilon_{ij}}{\frac{q_i q_j}{Dr_{ij}}} \cdot \frac{r_0}{r_{HX}}^{12} - \cdot \cdot \cdot \frac{\epsilon_{ij}}{\frac{r_0}{r_{HX}}}^{10}$$

- \mathcal{D}
- ω
- χ
- \mathcal{N}
- \mathcal{H}
- r_{HX}
- r_{ij}
- i, j
- i, j, k, l
- i, j, k, l
- q_i
- $U_{0_{ijkl}}$
- n_{ijkl}
- F_{ij}
- ϵ_{ij}
- r_0
- D

where E is the energy function, \mathcal{D} is the set of observed contacts, \mathcal{N} is the set of non-crossing contacts, \mathcal{H} is the set of crossing contacts, r_{HX} is the distance between the two ends of the molecule, r_{ij} is the distance between residues i and j , i, j, k, l are four residues forming a square contact, q_i is the charge of residue i , $U_{0_{ijkl}}$ is the energy of a square contact between residues i, j, k, l , n_{ijkl} is the number of times that a square contact between residues i, j, k, l has been observed, F_{ij} is the force between residues i and j , ϵ_{ij} is the spring constant between residues i and j , r_0 is the equilibrium distance between residues i and j , and D is the dielectric constant.

The energy function E is composed of three parts: a square contact term, a non-crossing contact term, and a crossing contact term. The square contact term is proportional to the number of observed contacts and the energy of each contact. The non-crossing contact term is proportional to the number of non-crossing contacts and the energy of each contact. The crossing contact term is proportional to the number of crossing contacts and the energy of each contact. The energy of each contact is determined by the distance between the two residues, the force between the two residues, the spring constant between the two residues, and the dielectric constant.

A.3.2 Other Prediction Methods.

There are many other prediction methods available, such as homology modeling, sequence-structure alignment, and neural networks. Homology modeling is based on the principle that similar proteins have similar structures. Sequence-structure alignment is based on the principle that similar sequences have similar structures. Neural networks are machine learning models that can learn from data to predict protein structures.

Homology modeling is a common method for predicting protein structures. It involves finding a template protein with a known structure that is similar to the target protein. The template structure is then used to build a model of the target protein's structure. Sequence-structure alignment is another common method for predicting protein structures. It involves aligning the target protein's sequence with the sequence of a template protein. The aligned sequences are then used to build a model of the target protein's structure. Neural networks are also used for protein structure prediction. They are trained on a large dataset of protein structures and their corresponding sequences. Once trained, they can predict the structure of a new protein based on its sequence.

$$\begin{aligned}
E = & \sum_{(i,j) \in \mathcal{B}} \frac{K_{r_{ij}}}{r_{ij} - r_{eq}}^2 \\
& + \sum_{(i,j,k) \in \mathcal{A}} \frac{K_{\Theta_{ijk}}}{r_{ijk} - r_{eq}}^2 \\
& + \sum_{(i,j,k,l) \in \mathcal{D}} \frac{K_{\Phi_{ijkl}}}{n_{ijkl} - n_{ijkl} - \gamma_{ijkl}}^2 \\
& - \sum_{(i,j) \in \mathcal{N}} \frac{\frac{A_{ij}}{r_{ij}}^{12} - \frac{B_{ij}}{r_{ij}}^6}{\frac{q_i q_j}{\pi \varepsilon r_{ij}}} \\
& - \sum_{(i,j) \in \mathcal{N}'} \frac{\frac{A_{ij}}{r_{ij}}^{12} - \frac{B_{ij}}{r_{ij}}^6}{\frac{q_i q_j}{\pi \varepsilon r_{ij}}}
\end{aligned}$$

- \mathcal{B}
- \mathcal{A}
- \mathcal{D}
- \mathcal{N}
- \mathcal{N}'
- r_{ij} i j
- r_{ijk} $i, j,$ k
- r_{ijkl} $i, j, k,$ l
- q_i i
- $K_{r_{ij}}$ r_{eq} $K_{\Theta_{ijk}}$ r_{eq} $K_{\Phi_{ijkl}}$ γ_{ijkl} A_{ij} B_{ij} ε

Simplification

Lattice

Lattice

Appendix B. Background on Genetic Algorithms

Genetic algorithms (GAs) are search procedures based on the principles of evolution and natural selection. A population of individuals is maintained, each representing a solution to a problem. The individuals are encoded in binary strings. The population undergoes several generations, during which it evolves through three main operators: selection, crossover, and mutation. Selection is used to choose individuals for reproduction based on their fitness. Crossover is used to combine the genetic material of two individuals to produce offspring. Mutation is used to introduce random changes in the genetic material of an individual. The process continues until a solution is found or a maximum number of generations is reached. GAs have been applied to a wide range of problems, including optimization, classification, and regression.

the first time, the term "evolution strategy" was used by Rechenberg [1973] to denote a stochastic optimization method based on the principles of natural selection and natural genetics. In the same year, Hinterding [1973] proposed a similar stochastic search algorithm based on the principles of natural selection and natural genetics. In 1975, Schaffer [1975] proposed a stochastic search algorithm based on the principles of natural selection and natural genetics. In 1975, Schaffer [1975] proposed a stochastic search algorithm based on the principles of natural selection and natural genetics. In 1975, Schaffer [1975] proposed a stochastic search algorithm based on the principles of natural selection and natural genetics.

B.1 Brief History of Evolutionary Algorithms

The first evolutionary algorithm was proposed by Holland [1975] in 1975. This algorithm was called "adaptive search" and it was based on the principles of natural selection and natural genetics.

In 1975, Schaffer [1975] proposed a stochastic search algorithm based on the principles of natural selection and natural genetics. In 1975, Schaffer [1975] proposed a stochastic search algorithm based on the principles of natural selection and natural genetics.

Evolutionstrategie

The first evolutionary algorithm was proposed by Holland [1975] in 1975. This algorithm was called "adaptive search" and it was based on the principles of natural selection and natural genetics.

In 1975, Schaffer [1975] proposed a stochastic search algorithm based on the principles of natural selection and natural genetics. In 1975, Schaffer [1975] proposed a stochastic search algorithm based on the principles of natural selection and natural genetics.

In 1975, Schaffer [1975] proposed a stochastic search algorithm based on the principles of natural selection and natural genetics. In 1975, Schaffer [1975] proposed a stochastic search algorithm based on the principles of natural selection and natural genetics.

In 1975, Schaffer [1975] proposed a stochastic search algorithm based on the principles of natural selection and natural genetics. In 1975, Schaffer [1975] proposed a stochastic search algorithm based on the principles of natural selection and natural genetics.

In 1975, Schaffer [1975] proposed a stochastic search algorithm based on the principles of natural selection and natural genetics. In 1975, Schaffer [1975] proposed a stochastic search algorithm based on the principles of natural selection and natural genetics.

In 1975, Schaffer [1975] proposed a stochastic search algorithm based on the principles of natural selection and natural genetics. In 1975, Schaffer [1975] proposed a stochastic search algorithm based on the principles of natural selection and natural genetics.

In 1975, Schaffer [1975] proposed a stochastic search algorithm based on the principles of natural selection and natural genetics. In 1975, Schaffer [1975] proposed a stochastic search algorithm based on the principles of natural selection and natural genetics.

In 1975, Schaffer [1975] proposed a stochastic search algorithm based on the principles of natural selection and natural genetics. In 1975, Schaffer [1975] proposed a stochastic search algorithm based on the principles of natural selection and natural genetics.

In 1975, Schaffer [1975] proposed a stochastic search algorithm based on the principles of natural selection and natural genetics. In 1975, Schaffer [1975] proposed a stochastic search algorithm based on the principles of natural selection and natural genetics.

In 1975, Schaffer [1975] proposed a stochastic search algorithm based on the principles of natural selection and natural genetics. In 1975, Schaffer [1975] proposed a stochastic search algorithm based on the principles of natural selection and natural genetics.

In 1975, Schaffer [1975] proposed a stochastic search algorithm based on the principles of natural selection and natural genetics. In 1975, Schaffer [1975] proposed a stochastic search algorithm based on the principles of natural selection and natural genetics.

In 1975, Schaffer [1975] proposed a stochastic search algorithm based on the principles of natural selection and natural genetics. In 1975, Schaffer [1975] proposed a stochastic search algorithm based on the principles of natural selection and natural genetics.

$$\mathbf{x}^{t+1} = \mathbf{x}^t + N(-\mu, \sigma)$$

$$\sigma_{\perp}$$

B.2 Origins of Genetic Algorithms

“Adaptation in Natural and Artificial Systems”
Schema Theorem Fundamental Theorem of Genetic Algorithms

B.3 Simple Genetic Algorithm (SGA)

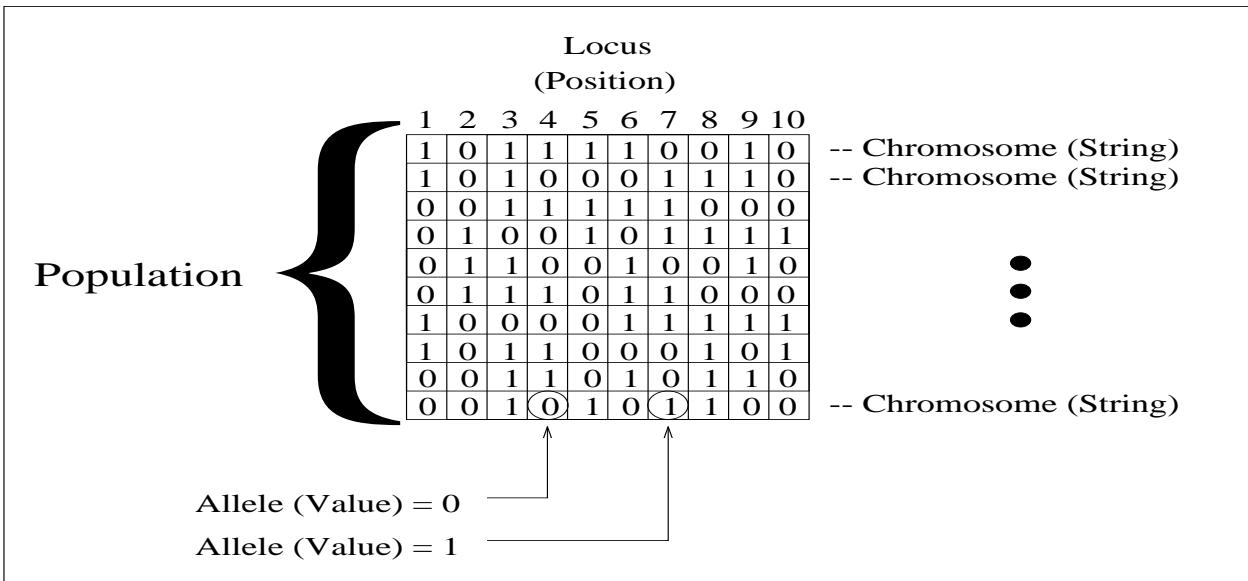
string chromosome genes

locus allele

¹ The term **phenotype** refers to the traits expressed by an individual, in this case the value returned by a function. Contrast this with **genotype** which refers to the traits that define the individual, for example the parameters of the function

evolves

population



x_{min}

x_{max}

x_{min}

x_{max}

B.3.1 Simple Genetic Algorithm Operators.

selection crossover mutation

selected

Crossover

converge

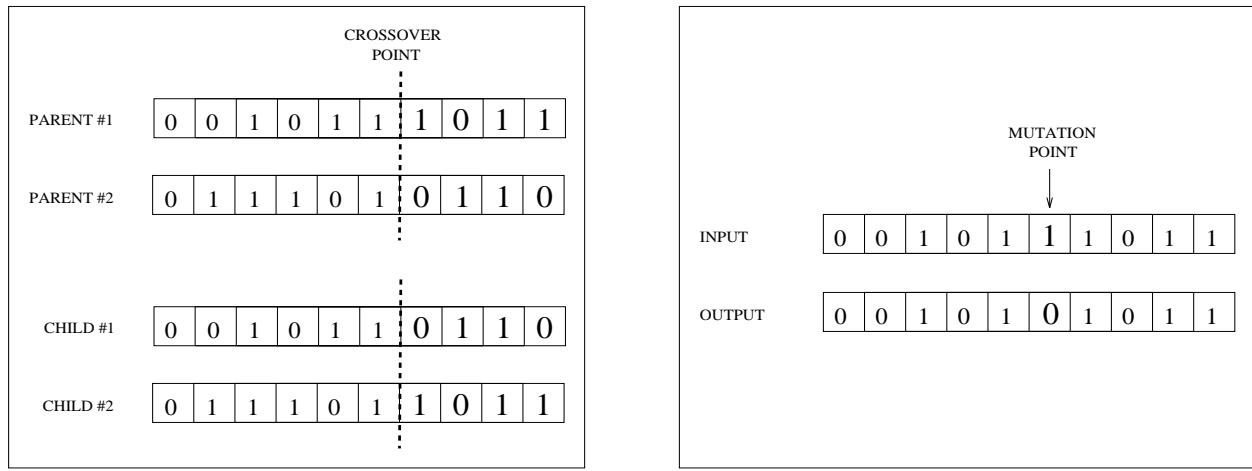
Mutation

single-point crossover bitwise mutation

Single-point crossover and bitwise mutation are two common genetic operators used in genetic algorithms. They both involve manipulating binary strings.

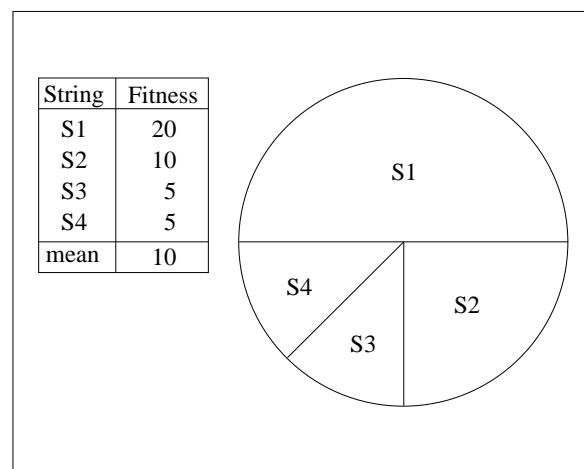
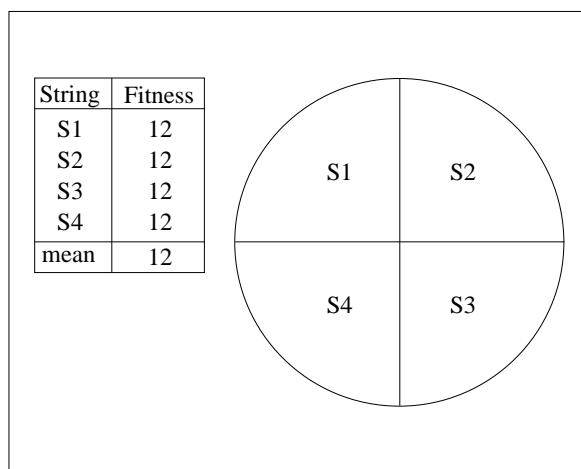
Single-point crossover: This operator creates two new offspring from two parent strings by exchanging a portion of each string starting from a randomly selected crossover point. In the diagram, Parent #1 and Parent #2 are crossed over at the 6th bit position. Child #1 receives the first 6 bits from Parent #1 and the last 4 bits from Parent #2. Child #2 receives the first 6 bits from Parent #2 and the last 4 bits from Parent #1.

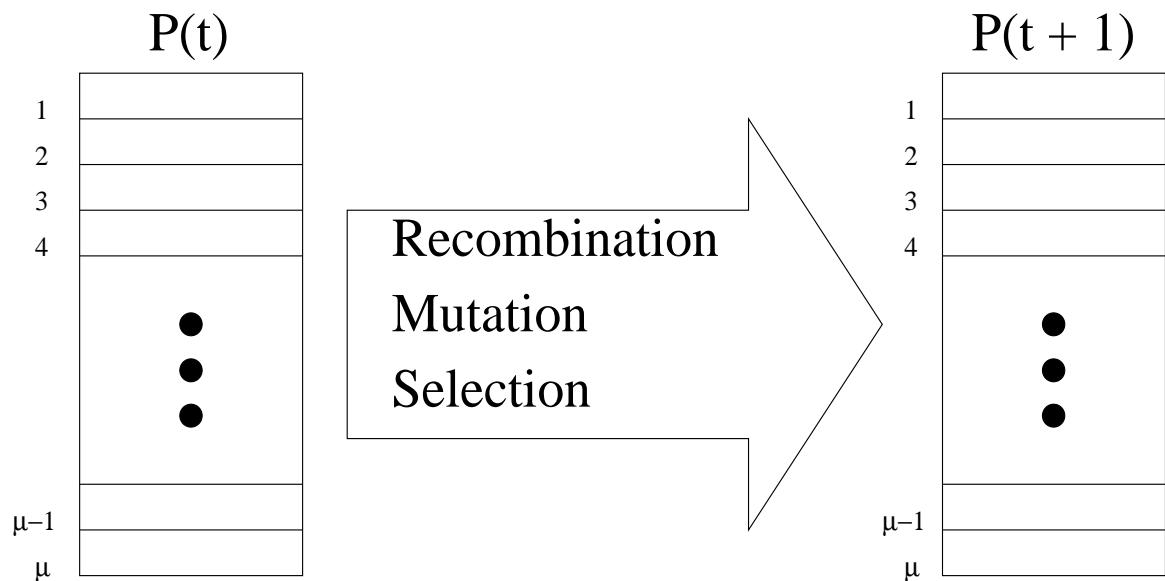
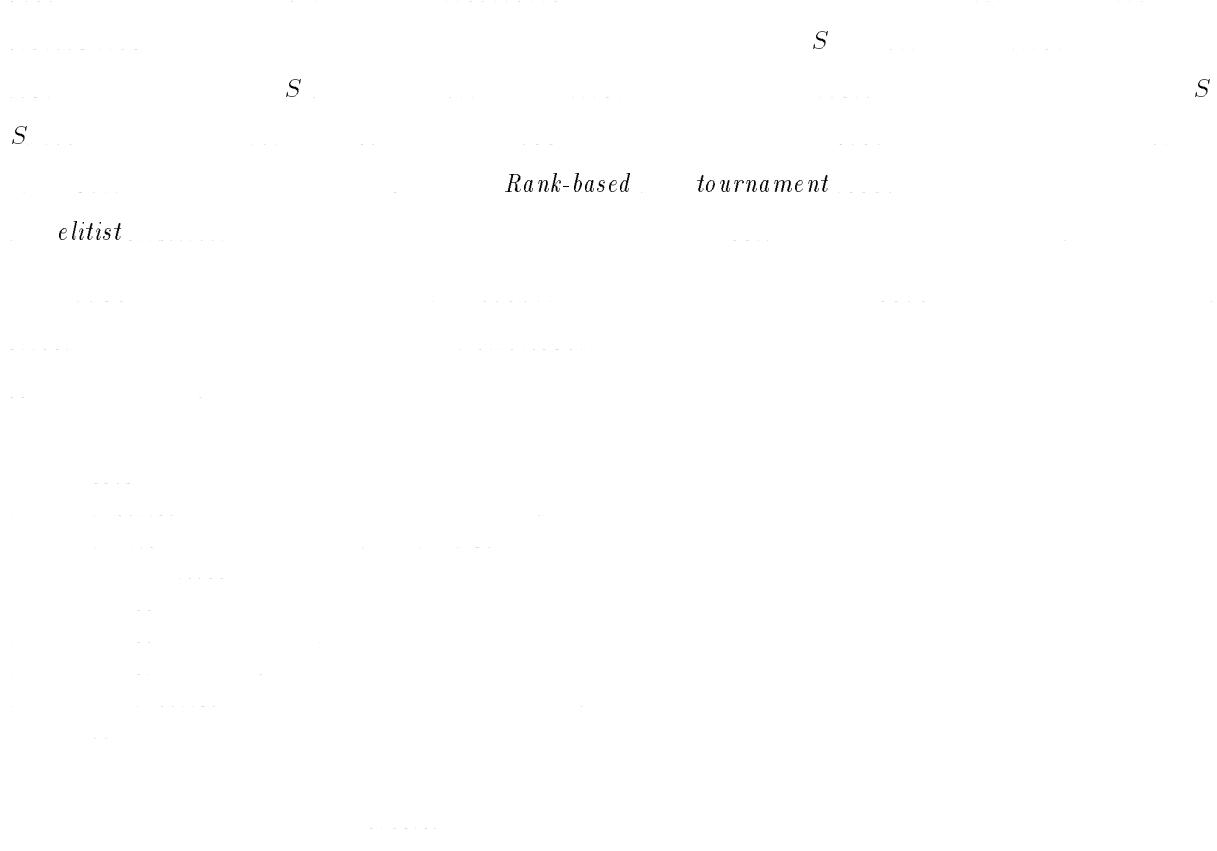
Bitwise mutation: This operator changes individual bits in a string. An input string is mutated at a specific point to produce an output string. In the diagram, the 5th bit of the input string (0, 0, 1, 0, 1, 1, 1, 0, 1, 1) is flipped from 0 to 1, resulting in the output string (0, 0, 1, 0, 1, 1, 0, 1, 0, 1).



proportional roulette-wheel

Proportional and roulette-wheel selection are methods for selecting individuals from a population based on their fitness.





B.3.2 Simple Genetic Algorithm Parameters.

Simple Genetic Algorithms have three main parameters: population size, mutation rate, and crossover rate. These parameters are typically set by the user or determined through experimentation. The population size is the number of individuals in the population. The mutation rate is the probability that a gene will change during reproduction. The crossover rate is the probability that two parents will produce offspring. These parameters can be adjusted to improve the performance of the algorithm. For example, increasing the population size can lead to better performance, but it also requires more computation time. On the other hand, increasing the mutation rate can help the algorithm explore the search space, but it can also lead to slower convergence. The crossover rate also affects the performance of the algorithm, but it is less critical than the other two parameters. In general, the performance of a Simple Genetic Algorithm depends on the specific problem being solved and the values of its parameters.

Simple Genetic Algorithms are a type of optimization algorithm that uses a population of individuals to find the best solution to a problem. The population consists of a set of individuals, each represented by a string of genes. The genes represent the variables of the problem. The population is initialized with random individuals. Then, the algorithm iterates through several generations. In each generation, the individuals are evaluated based on their fitness. The fittest individuals are selected to produce offspring. The offspring are created by combining the genes of the selected individuals. This process is called crossover. After crossover, some of the genes in the offspring may change due to mutation. The mutation rate is a parameter that controls the probability of mutation. Finally, the new population is formed by combining the original population and the offspring. This process continues until a stopping criterion is met, such as a maximum number of generations or a minimum fitness value. The final population contains the best solution found by the algorithm.

B.3.3 Mathematical Theory of How (Why) Simple GAs Work. Schemata

Simple Genetic Algorithms work by maintaining a population of individuals and applying selection, crossover, and mutation operators to produce offspring. The selection operator selects individuals based on their fitness. The crossover operator combines the genes of two parents to produce offspring. The mutation operator changes the genes of an individual. Schemata are patterns of genes that are used to represent the population. Schemata are defined by a template and a defining length. The template is a sequence of genes, where some positions are fixed and some are variable. The defining length is the length of the template. Schemata are used to represent the population because they allow for efficient search. By maintaining a population of schemata, the algorithm can focus on the most promising parts of the search space. The population is initialized with random schemata. Then, the algorithm iterates through several generations. In each generation, the schemata are evaluated based on their fitness. The fittest schemata are selected to produce offspring. The offspring are created by combining the genes of the selected schemata. This process is called crossover. After crossover, some of the genes in the offspring may change due to mutation. The mutation rate is a parameter that controls the probability of mutation. Finally, the new population is formed by combining the original population and the offspring. This process continues until a stopping criterion is met, such as a maximum number of generations or a minimum fitness value. The final population contains the best solution found by the algorithm.

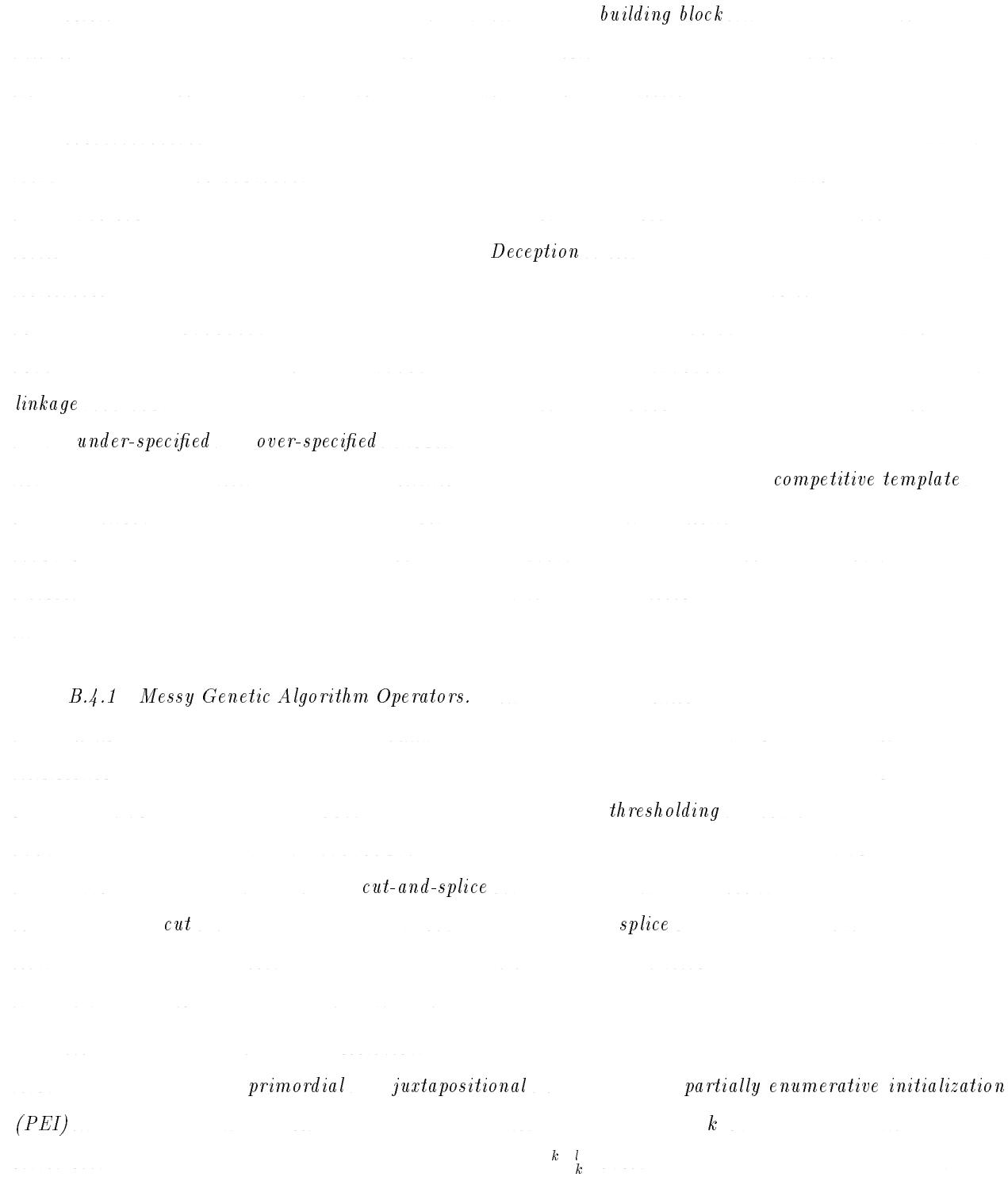
$$m(H, t) \geq m(H, t) \cdot \frac{f(H)}{f} - p_c \frac{\delta(H)}{l-1} - o(H) p_m,$$

H is a $n \times l$ matrix with entries from $\{0, 1\}$.
 $m(H, t)$ is the number of columns of H with weight t .
 $f(H)$ is the number of columns of H with weight 1 .
 $\delta(H)$ is the number of columns of H with weight 0 .
 p_m is the probability of a column of H having weight 0 .
 p_c is the probability of a column of H having weight 1 .
 $o(H)$ is the number of columns of H with weight > 1 .
 l is the number of columns of H .
 $implicit parallelism$

B.3.3.1 Complexity Analysis.

$$\mathcal{O}(nl) = n \cdot l = \mathcal{O}(nl)$$

B.4 Messy Genetic Algorithm (*mGA*)



the population size, the number of crossover points, and the mutation rate.

For each parameter, we conducted a series of experiments to determine its effect on the performance of the algorithm.

We used a two-factor factorial design to study the interaction between the parameters.

The factors were the population size and the number of crossover points.

The levels of the population size were 10, 20, 30, and 40.

The levels of the number of crossover points were 1, 2, 3, and 4.

The results showed that the population size had a significant effect on the performance of the algorithm.

The performance of the algorithm increased as the population size increased.

The performance of the algorithm decreased as the number of crossover points increased.

The performance of the algorithm was best when the population size was 20 and the number of crossover points was 2.

The performance of the algorithm was worst when the population size was 40 and the number of crossover points was 4.

The performance of the algorithm was intermediate when the population size was 10 or 30 and the number of crossover points was 1 or 3.

The performance of the algorithm was intermediate when the population size was 20 and the number of crossover points was 1 or 3.

The performance of the algorithm was intermediate when the population size was 30 and the number of crossover points was 1 or 3.

The performance of the algorithm was intermediate when the population size was 40 and the number of crossover points was 1 or 3.

The performance of the algorithm was intermediate when the population size was 10 or 30 and the number of crossover points was 2 or 4.

The performance of the algorithm was intermediate when the population size was 20 and the number of crossover points was 2 or 4.

The performance of the algorithm was intermediate when the population size was 30 and the number of crossover points was 2 or 4.

The performance of the algorithm was intermediate when the population size was 40 and the number of crossover points was 2 or 4.

The performance of the algorithm was intermediate when the population size was 10 or 30 and the number of crossover points was 1 or 3.

The performance of the algorithm was intermediate when the population size was 20 and the number of crossover points was 1 or 3.

The performance of the algorithm was intermediate when the population size was 30 and the number of crossover points was 1 or 3.

The performance of the algorithm was intermediate when the population size was 40 and the number of crossover points was 1 or 3.

The performance of the algorithm was intermediate when the population size was 10 or 30 and the number of crossover points was 2 or 4.

The performance of the algorithm was intermediate when the population size was 20 and the number of crossover points was 2 or 4.

The performance of the algorithm was intermediate when the population size was 30 and the number of crossover points was 2 or 4.

The performance of the algorithm was intermediate when the population size was 40 and the number of crossover points was 2 or 4.

The performance of the algorithm was intermediate when the population size was 10 or 30 and the number of crossover points was 1 or 3.

The performance of the algorithm was intermediate when the population size was 20 and the number of crossover points was 1 or 3.

The performance of the algorithm was intermediate when the population size was 30 and the number of crossover points was 1 or 3.

The performance of the algorithm was intermediate when the population size was 40 and the number of crossover points was 1 or 3.

B.4.2 Messy Genetic Algorithm Parameters.

B.4.3 Mathematical Theory of How (Why) Messy GAs Work.

the normalized expected defining length is

$$\frac{\langle \delta \rangle}{l+1} = k$$

so the normalized expected defining length is

$$\frac{\langle \delta \rangle}{l} = \frac{k-1}{k}$$

so the normalized expected defining length is

$$\frac{\langle \delta \rangle}{l} = \frac{k-1}{k}$$

B.4.3.1 Complexity Analysis.

$$\mathcal{O}(l^k)$$
$$\mathcal{O}(l^{k-1})$$
$$n \ll much$$

B.5 Fast Messy Genetic Algorithm (fmGA)

the fast messy genetic algorithm (fmGA) is a probabilistically complete initialization

B.5.1 Fast Messy Genetic Algorithm Operators.

probabilistically complete initialization

P. 5/9 – Exact Message-Genetic Algorithm Parameters

B.5.2 Fast Messy Genetic Algorithm Parameters.

B.5.2 Mathematical Theory of Heavy (W^L) Event Measures. Cf. 5.2.3. Weak

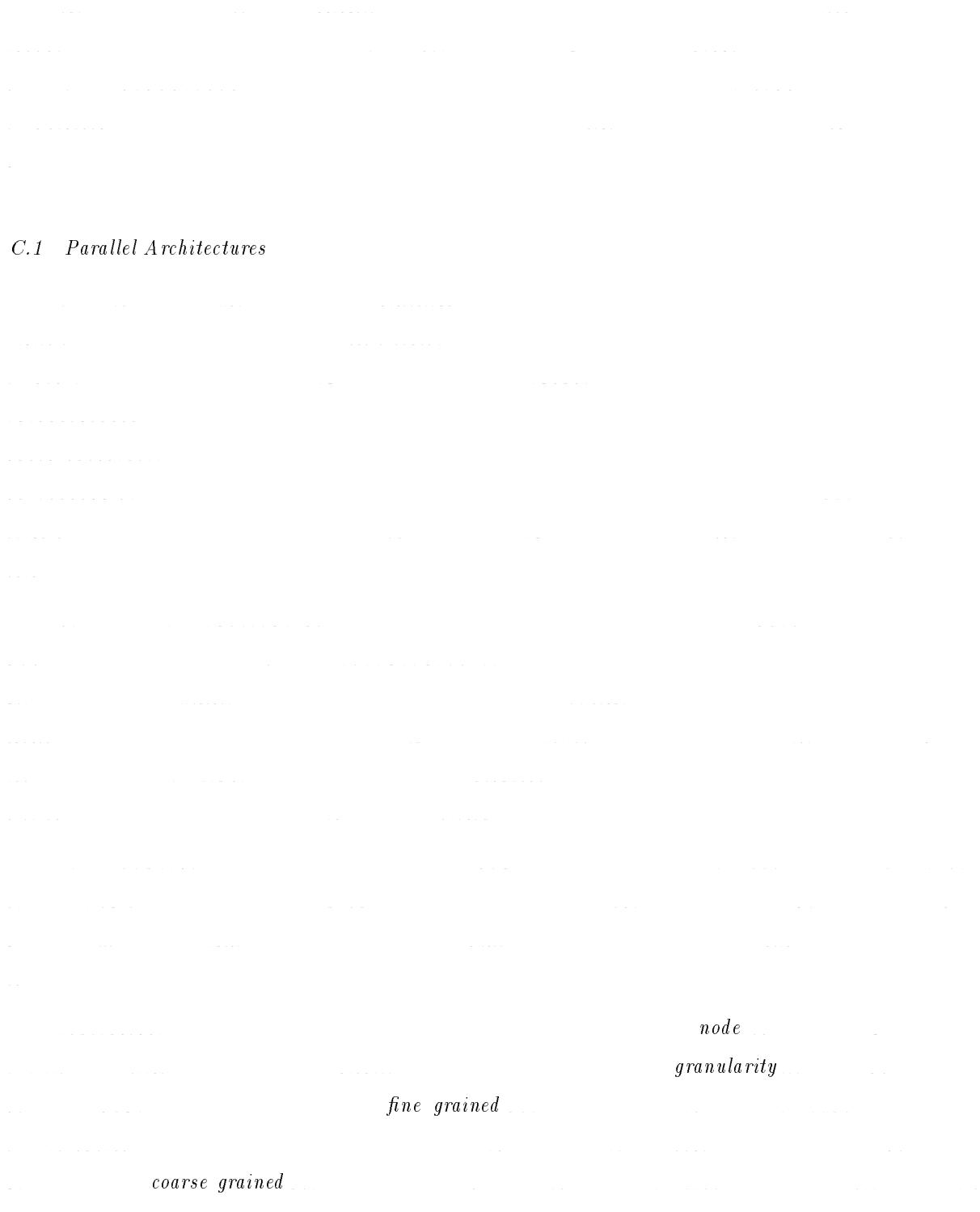
B.5.3.1 Complexity Analysis.

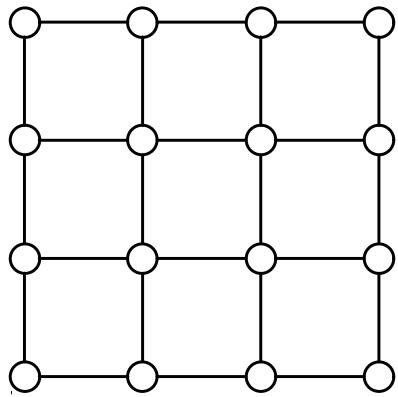
$$\mathcal{O}(l) = l$$

$$\mathcal{O}(l - l) = \mathcal{O}(l^k)$$

$$\mathcal{O}(nl)$$

Appendix C. Background on Parallel Computing





\times *interconnection topology*

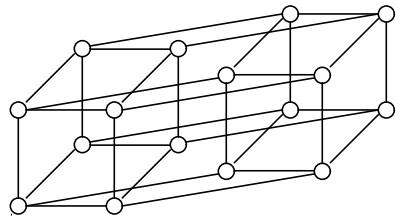
shared memory *distributed memory*

message passing *interconnection topology* *network*

\times *dimension N*

hypercube *2-D mesh*

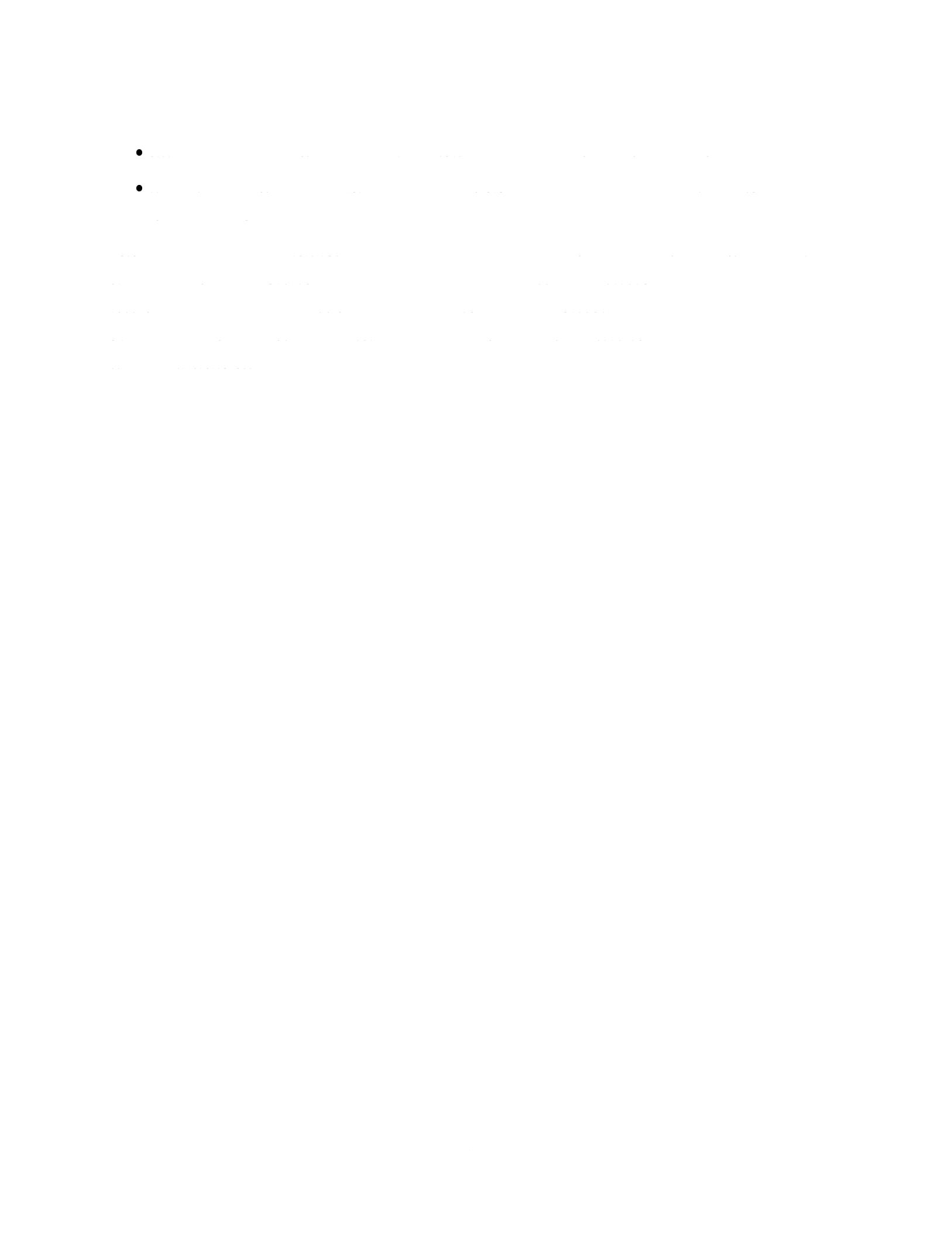
interconnection topology *message passing*



C.2 Parallel Algorithms.

- A parallel algorithm is an algorithm designed to be executed on a parallel computer. It typically involves dividing the computation into multiple tasks that can be executed simultaneously on different processors.
- In a parallel algorithm, the data is often partitioned among the processors. This allows each processor to work on a portion of the data independently, which can significantly reduce the overall execution time.
- Communication between processors is an important consideration in parallel algorithms. Data must be exchanged between processors to ensure that all parts of the computation are synchronized and that the final results are correct.
- There are several types of parallel algorithms, including:

 - **MapReduce**: A programming model for processing large datasets in parallel. It consists of two main stages: map and reduce. In the map stage, data is processed by mappers to produce intermediate key-value pairs. These pairs are then grouped by key and processed by reducers to produce the final output.
 - **MapReduce**: A programming model for processing large datasets in parallel. It consists of two main stages: map and reduce. In the map stage, data is processed by mappers to produce intermediate key-value pairs. These pairs are then grouped by key and processed by reducers to produce the final output.
 - **MapReduce**: A programming model for processing large datasets in parallel. It consists of two main stages: map and reduce. In the map stage, data is processed by mappers to produce intermediate key-value pairs. These pairs are then grouped by key and processed by reducers to produce the final output.



Appendix D. PHGA Operation

```
something
in    in.something
something
in
in.something
in.something
> /dev/null
psga_param
psga_default psga_param
psga_default
in
mpirun
-np
-sz
in.2.10.24
&
> /dev/null
```

```
Experiments = 1
Total Trials = 500
Population Size = 20
Structure Length = 240
Crossover Rate = 0.65
Mutation Rate = 0.005
Generation Gap = 1.0
Scaling Window = 1
Report Interval = 1
Structures Saved = 1
Max Gens w/o Eval = 10
Dump Interval = 0
Dumps Saved = 0
Options = ycel
Number of Peaks = 1.0
Minimization Prob = 1.0
Replacement Prob = 1.0
Random Seed = 987654321
Rank Min = 1.5
```

Appendix E. Genocop-III

E.1 Algorithm

Procedure Genocop III

```
begin
     $t \leftarrow t$ 
     $P_s t$ 
     $P_r t$ 
     $P_s t$ 
     $P_r t$ 
    while not do
        begin
             $t \leftarrow t$ 
             $P_s t$ 
             $P_s t$ 
             $P_s t$ 
            if  $t \bmod k$  then
                begin
                     $P_r t$ 
                     $P_r t$ 
                     $P_r t$ 
                end
        end
    end
end
```

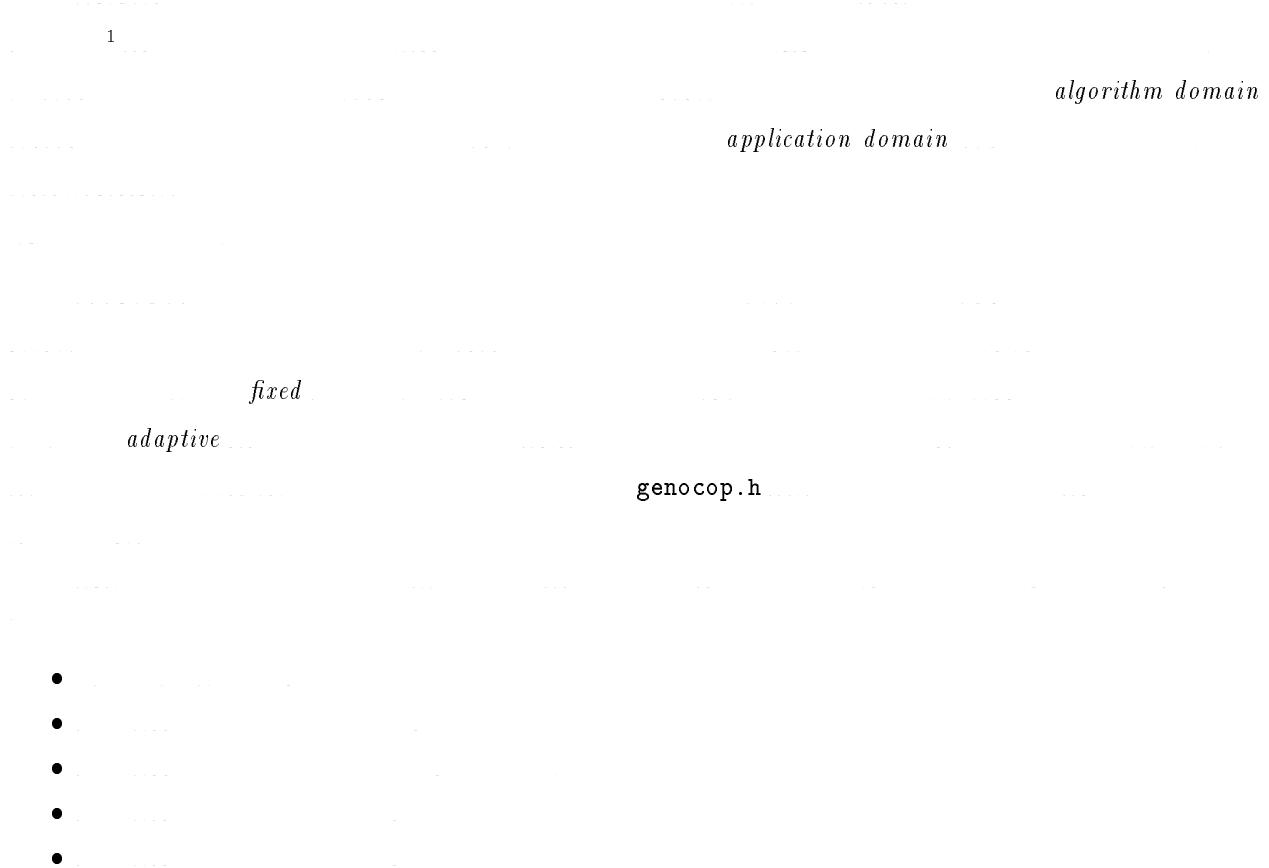
```

procedure evaluate  $P_s$   $t$ 
begin
     $\vec{s} \in P_s$   $t$  do
        if  $\vec{s} \in \mathcal{F}$ 
        then  $\vec{s} = f(\vec{s})$  else
        begin
             $\vec{r} \in P_r$   $t$ 
             $\vec{z} \in \mathcal{F}$ 
             $\vec{s} = f(\vec{z})$ 
            if  $f(\vec{r}) > f(\vec{z})$  then
                 $\vec{r} = \vec{z} = P_r$ 
                 $\vec{s} = \vec{z} = P_s$ 
             $p_r$ 
        end
    end
end

```

P_s

E.2 Input Parameters



¹ Here the phrase “Domain Constraints” is used in a context more limited than normally used in computer science, specifically, the allowable range of specified variables. If a domain constraint is not defined for a variable, it defaults to the architecture dependent range for R .

| | | |
|---|--|-------------------|
| | | |
| - | | $N \cup$ |
| - | | N |
| - | | R |
| | | $\leq iSeed \leq$ |
| | | $\leq iSeed \leq$ |

- $\leq iSeed \leq$ is a condition that must be met for the current node to be considered for expansion.
- $\leq iSeed \leq$ is a condition that must be met for the current node to be considered for expansion.
- $\leq iSeed \leq$ is a condition that must be met for the current node to be considered for expansion.
- $\leq iSeed \leq$ is a condition that must be met for the current node to be considered for expansion.
- $\leq iSeed \leq$ is a condition that must be met for the current node to be considered for expansion.

E.3 Operators

E.3.1 Whole arithmetical crossover.

$$\vec{x} = \vec{y} \quad \text{and} \quad \vec{z} = a\vec{x} + (1-a)\vec{y}$$

E.3.2 Simple arithmetical crossover.

$$\vec{x} = \vec{y} \quad \text{and} \quad \vec{z} = a\vec{x} + (1-a)\vec{y}$$

E.3.3 Whole uniform mutation.

$$\vec{x}' = x_1, \dots, x_k', \dots, x_n \quad \text{and} \quad x_k' = \begin{cases} x_k & \text{if } k \neq \text{left } k \\ \Delta(t, r) & \text{if } k = \text{left } k \\ x_k & \text{if } k \neq \text{right } k \end{cases}$$

E.3.4 Boundary mutation.

$$x_k^{t+1} = \begin{cases} x_k^t & \text{if } k \neq \text{left } k \text{ or } k \neq \text{right } k \\ \Delta(t, r) & \text{if } k = \text{left } k \\ x_k^t - \Delta(t, r) & \text{if } k = \text{right } k \end{cases}$$

E.3.5 Non-uniform mutation.

$$x_k^{t+1} = \begin{cases} x_k^t & \text{if } k \neq \text{left } k \text{ or } k \neq \text{right } k \\ \Delta(t, y) & \text{if } k = \text{left } k \\ x_k^t - \Delta(t, y) & \text{if } k = \text{right } k \end{cases}$$

E.3.6 Non-uniform mutation.

$$x_k^{t+1} = \begin{cases} x_k^t & \text{if } k \neq \text{left } k \text{ or } k \neq \text{right } k \\ \Delta(t, y) & \text{if } k = \text{left } k \\ x_k^t - \Delta(t, y) & \text{if } k = \text{right } k \end{cases}$$

E.3.7 Non-uniform mutation.

$$x_k^{t+1} = \begin{cases} x_k^t & \text{if } k \neq \text{left } k \text{ or } k \neq \text{right } k \\ \Delta(t, y) & \text{if } k = \text{left } k \\ x_k^t - \Delta(t, y) & \text{if } k = \text{right } k \end{cases}$$

E.3.8 Non-uniform mutation.

$$x_k^{t+1} = \begin{cases} x_k^t & \text{if } k \neq \text{left } k \text{ or } k \neq \text{right } k \\ \Delta(t, y) & \text{if } k = \text{left } k \\ x_k^t - \Delta(t, y) & \text{if } k = \text{right } k \end{cases}$$

E.3.9 Non-uniform mutation.

$$x_k^{t+1} = \begin{cases} x_k^t & \text{if } k \neq \text{left } k \text{ or } k \neq \text{right } k \\ \Delta(t, y) & \text{if } k = \text{left } k \\ x_k^t - \Delta(t, y) & \text{if } k = \text{right } k \end{cases}$$

E.3.6 Whole non-uniform mutation.

Whole non-uniform mutation is a variation operator that generates a new solution by applying a non-uniform mutation to an existing solution. This operator is typically used in genetic algorithms to maintain diversity in the population.

E.3.7 Heuristic crossover.

heuristic crossover

Heuristic crossover is a variation operator that generates a new solution by combining two existing solutions using a heuristic rule. This operator is typically used in genetic algorithms to explore the search space more effectively.

$$\vec{x} \quad \vec{y} \quad \vec{z} = r \cdot \vec{x} + (1-r) \cdot \vec{y}$$

$$f(\vec{x}) \leq f(\vec{y}) \quad f(\vec{z}) \geq f(\vec{y})$$

E.3.8 Gaussian mutation.

Gaussian mutation

Gaussian mutation is a variation operator that generates a new solution by adding a Gaussian noise vector to an existing solution. This operator is typically used in genetic algorithms to maintain diversity in the population.

$$\vec{x}^{t+1} = \vec{x}^t + N(0, \vec{\sigma})$$

$$N(0, \vec{\sigma})$$

$$\vec{\sigma}$$

$$\vec{\sigma}$$

E.3.9 Pool recombination operator.

Pool recombination operator is a variation operator that generates a new solution by combining multiple existing solutions from a pool. This operator is typically used in genetic algorithms to maintain diversity in the population.

E.3.10 Scatter search operator.

Scatter search operator is a variation operator that generates a new solution by combining multiple existing solutions from a pool. This operator is typically used in genetic algorithms to maintain diversity in the population.

$$k > J - w$$

$$\vec{b} = \vec{w}$$

$$\vec{c} = \vec{x}_i / k$$

$$\vec{x}_i \in J - \vec{w}$$

$$\vec{y} = \vec{c} + \vec{w}$$

orgy

Appendix F. Statistical Methods

F.1 Analysis of Variance (ANOVA)

F.1.1 Single Factor Factorial Design

| |
|--|
| $y_{ij} = \mu + \tau_i + \epsilon_{ij}$ <p style="text-align: center;">$i = 1, \dots, a$ $j = 1, \dots, n$</p> <p style="text-align: right;">τ_i overall mean τ_i μ treatment effect ϵ_{ij}</p> |
| σ^2 |
| τ_i fixed effects model τ_i only random samples |
| $SS_T = \sum_{i=1}^a \sum_{j=1}^n (y_{ij} - \bar{y}_{..})^2$ $N = a \cdot n$ $\frac{a}{i=1} n_i$ $\bar{y}_{..}$ <p style="text-align: center;">analysis of variance</p> |

ANOVA table

| Source of Variation | | Sum of Squares (SS) | Degrees of Freedom (DF) | Mean Square (MS) | F-value | P-value |
|-----------------------|--|----------------------|-------------------------|----------------------|---------|---------|
| Between Groups (A) | | SS_A | $df_A = a - 1$ | $MS_A = SS_A / df_A$ | | |
| Within Groups (Error) | | SS_E | $df_E = N - a$ | $MS_E = SS_E / df_E$ | | |
| | | $SS_T = SS_A + SS_E$ | $df_T = N - 1$ | $MS_T = SS_T / df_T$ | | |

ANOVA table

| Source of Variation | | Sum of Squares (SS) | Degrees of Freedom (DF) | Mean Square (MS) | F-value | P-value |
|-----------------------|--|----------------------|-------------------------|----------------------|---------|---------|
| Between Groups (A) | | SS_A | $df_A = a - 1$ | $MS_A = SS_A / df_A$ | | |
| Within Groups (Error) | | SS_E | $df_E = N - a$ | $MS_E = SS_E / df_E$ | | |
| | | $SS_T = SS_A + SS_E$ | $df_T = N - 1$ | $MS_T = SS_T / df_T$ | | |

$$SS_T = SS_{Treatments} + SSE$$

$$SS_{Treatments} = \sum_{i=1}^a \tau_i^2 - \frac{(\tau_1 + \tau_2 + \dots + \tau_a)^2}{N-a}$$

$$no\; difference \qquad \qquad H_0: \mu_1 = \mu_2 = \dots = \mu_a$$

$$H_0: \tau_1 = \tau_2 = \dots = \tau_a$$

$$F_0 = \frac{SS_{Treatment}/(a-1)}{SS_E/(N-a)} = \frac{MS_{Treatments}}{MS_E}$$

$$F_0 = \frac{(a-1)SST}{(N-a)SSE} \sim F_{a-1, N-a}$$

$$F_0 > F_{\alpha, a-1, N-a}$$

$$F_0 = \frac{MS_{Treatments}}{MS_E} = \frac{\sum_{i=1}^a \frac{y_i^2}{n} - \frac{\sum_{i=1}^a y_i^2}{N}}{\sum_{i=1}^a \frac{y_i^2}{n} - \frac{\sum_{i=1}^a y_i^2}{N}}$$

$$\begin{aligned} SST &= \sum_{i=1}^a \sum_{j=1}^n y_{ij}^2 - \frac{\sum_{i=1}^a y_i^2}{N} \\ SS_{Treatments} &= \sum_{i=1}^a \frac{y_i^2}{n} - \frac{\sum_{i=1}^a y_i^2}{N} \\ SSE &= SST - SS_{Treatments} \end{aligned}$$

$$y$$

| | | | | $F_0 = \frac{MS_{Treatments}}{MS_E}$ |
|--|-------------------|-------|-------------------|--------------------------------------|
| | $SS_{Treatments}$ | $a-1$ | $MS_{Treatments}$ | $F_0 = \frac{MS_{Treatments}}{MS_E}$ |
| | SSE | $N-a$ | MS_E | |
| | SST | $N-1$ | | |

$$y_i = \frac{y_{ij}}{\sqrt{n}}, \quad y_i = y_i/n, \quad i = 1, \dots, a$$

$$y_{..} = \frac{1}{a \times b} \sum_{i=1}^a \sum_{j=1}^b y_{ij}, \quad y_{..} = y_{..}/N$$

F.1.2 Two Factor Factorial Design

| | | | | | | | | | |
|-----|-------|-----------|---|------------------|-------------------|------------------|------------------|-------------------|-------------------|
| | a | A | b | B | | | | | |
| | | | | | $i = 1, \dots, a$ | | | | |
| | | y_{ijk} | μ | τ_i | β_j | $\tau\beta_{ij}$ | ϵ_{ijk} | $j = 1, \dots, b$ | $k = 1, \dots, n$ |
| | | | | | | | | | |
| | μ | | | τ_i | | i | | A | β_j |
| j | | | B | $\tau\beta_{ij}$ | | | | | ϵ_{ijk} |
| | | | | | | | | | |
| | | H_0 | $\tau_1 = \tau_2 = \dots = \tau_a$ | | | | | | |
| | | H_1 | $\exists \tau_i /$ | | | | | | |
| | | B | | | | | | | |
| | | H_0 | $\beta_1 = \beta_2 = \dots = \beta_b$ | | | | | | |
| | | H_1 | $\exists \beta_i /$ | | | | | | |
| | A | | B | | | | | | |
| | | H_0 | $\tau\beta_{ij} = 0 \quad \forall i, j$ | | | | | | |
| | | H_1 | $\exists \tau_i /$ | | | | | | |
| | | | H_0 | | | | | | |

| | | | | F_0 |
|--|-----------|------------|--------------------------------------|----------------------------|
| | SS_A | $a -$ | $MS_A \frac{SS_A}{a-1}$ | $F_0 \frac{MS_A}{MS_E}$ |
| | SS_B | $b -$ | $MS_B \frac{SS_B}{b-1}$ | $F_0 \frac{MS_B}{MS_E}$ |
| | SS_{AB} | $a - b -$ | $MS_{AB} \frac{SS_{AB}}{(a-1)(b-1)}$ | $F_0 \frac{MS_{AB}}{MS_E}$ |
| | SSE | $ab - n -$ | $MSE \frac{SSE}{ab(n-1)}$ | |
| | SS_T | $abn -$ | | |

$$\begin{aligned}
SS_T &= \sum_{i=1}^a \sum_{j=1}^b \sum_{k=1}^n y_{ijk}^2 - \frac{\bar{y}_{...}^2}{abn} \\
SS_A &= \sum_{i=1}^a \frac{\bar{y}_{i..}^2}{bn} - \frac{\bar{y}_{...}^2}{abn} \\
SS_B &= \sum_{j=1}^b \frac{\bar{y}_{.j.}^2}{an} - \frac{\bar{y}_{...}^2}{abn} \\
SS_{AB} &= SS_{Subtotals} - SS_A - SS_B \\
SS_E &= SS_T - SS_{Subtotals} \\
SS_{Subtotals} &= \sum_{i=1}^a \sum_{j=1}^b \frac{y_{ij.}^2}{n} - \frac{\bar{y}_{...}^2}{abn}
\end{aligned}$$

$$\begin{aligned}
y_{i..} &= \frac{y_{ijk}}{n}, \quad y_{i..} = y_{i..}/bn \quad i = 1, \dots, a \\
y_{.j.} &= \frac{y_{ijk}}{n}, \quad y_{.j.} = y_{.j.}/an \quad j = 1, \dots, b \\
y_{ij.} &= \frac{y_{ijk}}{n}, \quad y_{ij.} = y_{ij.}/n \quad i = 1, \dots, a \\
y_{..} &= \frac{y_{ijk}}{n}, \quad y_{..} = y_{..}/abn
\end{aligned}$$

F.2 Kruskal-Wallis H Test.

$$\begin{array}{ccccccc}
& & & & & k & \\
& & & & & & \\
& n & & & & k & \\
& & & & & & \\
& i & & & & & R_i \\
& & & & & &
\end{array}$$

Suppose we have k independent samples from k populations. We wish to test the null hypothesis

H_0 : the samples are from identical populations
against the alternative hypothesis

H_1 : the populations are not identical
at the α level of significance.

1. Compute h . Calculate

$$h = \frac{12}{n(n+1)} \sum_{i=1}^k \frac{R_i^2}{n_i} - 3(n+1)$$

2. Accept or reject H_0 . If $h > \chi_{k-1, \alpha}^2$, reject H_0 ; otherwise accept H_0 .

(1)

Probability, Statistics, and Queueing Theory: With Computer Science Applications

Journal of Global Optimization

Evolutionary Algorithms in Theory and Practice

Algorithmics: Theory and Practice

Journal of Computational Chemistry

*Proceedings
of the Fourth International Conference on Genetic Algorithms*

Physics Today

Parallel Program Design: A Foundation

*Journal of Global
Optimization*

*Grand Challenges 1993: High Per-
formance Computing and Communications*

Protein Folding

Proteins: Structures and Molecular Properties

Handbook of Genetic Algorithms

*International Conference
on Genetic Algorithms*

Proceedings of the Fifth Interantional Conference on Genetic Algorithms

An Analysis of the Behavior of a Class of Genetic Adaptive Systems

*IEEE Transactions on Systems,
Man and Cybernetics*

*Genetic Algorithms and
their Applications: Proceedings of the Second International Conference on Genetic Algorithms*

Parallel Genetic Algorithms

The Second Annual Conference on Evolutionary Programming

Task Scheduling in Parallel and Distributed Systems

Proceedings of the Third International Conference on Genetic Algorithms

Proceedings of the Fourth International Conference on Genetic Algorithms

The Mathematica Journal
International Conference on Genetic Algorithms

Proceedings of the Second IEEE Conference on Evolutionary Computation

The Second Annual Conference on Evolutionary Programming

Artificial Intelligence Through Simulated Evolution

IEEE—ACSSC

Foundations of Genetic Algorithms 2
Computers and Intractability: A Guide to the Theory of NP-Completeness

Decision Sciences

Genetic Algorithms in Search, Optimization, and Machine Learning

*Genetic Algorithms, Noise, and the Sizing
of Populations*

*Proceedings of
the Fifth International Conference on Genetic Algorithms*

Complex Systems

*International
Conference on Genetic Algorithms*

Complex Systems

Proceedings of the Second International Conference on Genetic Algorithms

Proceedings of the Fifth International Conference on Genetic Algorithms

*IEEE Transactions on
Systems, Man & Cybernetics*

Proceedings of the Third International Conference on Genetic Algorithms

Proceedings of the Fourth International Conference on Genetic Algorithms

Deception Considered Harmful

*Proceedings of International Parallel Processing
Symposium*

Adaptation in Natural and Artificial Systems

Scientific American

Principles of Biochemistry

Biochemistry



Applied Computing

1997: Proceedings of the 1997 Symposium on Applied Computing

The Origins of Order, Self-Organization and Selection in Evolution

Advances in Parallel Algorithms

Introduction to Parallel Computing:

Design and Analysis of Algorithms

Compendium of Parallel Programs for the Intel iPSC Computers

Calculus with Analytic Geometry

Journal of Global Optimization

The Protein Folding Problem and Tertiary Structure Prediction

Arbeitspapiere der

GMD 748

Real Time System Design

Introduction to Parallel Computing

Proceedings of the National Academy of Science USA

Theory and Problems of Organic Chemistry

Applied Computing 1996: Proceedings of the 1996 Symposium on Applied Computing

Evolutionary Computation

Proceedings of the 3rd Annual Conference on Evolutionary Programming

Evolutionary Computation Journal

Genetic Algorithms + Data Structures = Evolution Programs

Genetic Algorithms + Data Structures = Evolution Programs

Molecular Structure (Theochem)

Design and Analysis of Experiments

Parallel Computing

Journal of Computational Chemistry

SPIE

Proceedings 1

Heuristics

Proceedings of the Third International Conference on Genetic Algorithms

Numerical Recipes

in C: The Art of Scientific Computing

Protein Science

Third

International Conference on Genetic Algorithms

*Proceedings of the Fourth International Conference
on Genetic Algorithms*

MPI: The Complete

Reference

WEBSTER'S II New Riverside University Dictionary

IEEE-CH

*Proceedings of the Fourth Interna-
tional Conference on Genetic Algorithms*

COMPUTER

*Proceedings of the Fourth
International Conference on Genetic Algorithms*

Biochemistry

Proceedings of the Fourth

*Algorithms and Their Applications: Proceedings of the Second International Conference on Genetic
Algorithms*

Genetic

genitor
International Conference on Genetic Algorithms

PPSN III

Proceedings of the Fourth International Conference on Genetic Algorithms

Foundation of Genetic Algorithms

Vita